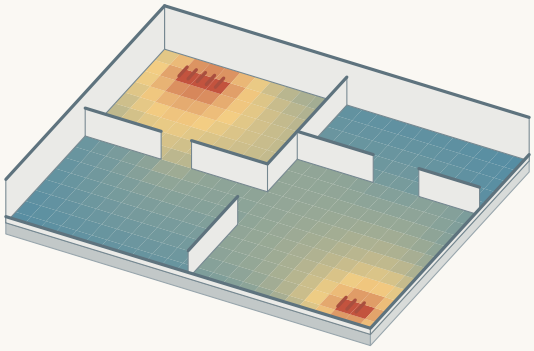
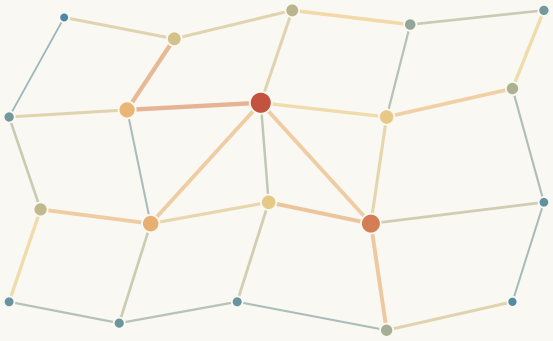
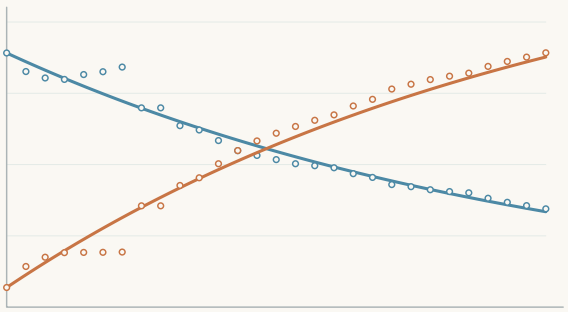
LECTURE NOTES

Numerical Simulation of Mathematical Models

Silvia Gazzola <silvia.gazzola@unipi.it>
Leonardo Robol <leonardo.robol@unipi.it>









Academic Year 2026–2027

Last updated on October 20, 2026

Contents

I. Modeling tools **11**

1. Solving linear systems and computing eigenvalues **13**
1.1. Sensitivity, error, and residual . . . 13
1.2. Direct methods . . . 14
1.3. Stationary iterations . . . 14
1.4. Krylov methods . . . 15
1.5. Preconditioning . . . 15
1.6. Least-squares problems . . . 15
1.7. A practical choice . . . 16
1.8. The singular value decomposition (SVD) . . . 16

2. Integration of ordinary differential equations **17**
2.1. Consistency and convergence . . . 17
2.2. Absolute stability and stiffness . . . 17
2.2.1. A simple stiff problem . . . 18
2.3. Runge–Kutta methods . . . 19
2.4. Linear multistep methods . . . 20
2.5. Solving implicit steps . . . 20
2.6. Error control and method selection . . . 21

3. Computing eigenvalues **23**
3.1. Sensitivity and residuals . . . 23
3.2. Dense problems: the QR algorithm . . . 23
3.3. Power and inverse iteration . . . 24
3.4. Krylov eigensolvers . . . 24
3.5. Jacobi–Davidson . . . 24
3.6. Generalized eigenvalue problems . . . 25

4. Optimization **27**
4.1. Formulation, notation, and existence . . . 27
4.2. First- and second-order optimality . . . 28
4.2.1. Convexity and global optimality . . . 28
4.3. Local models, descent directions, and line searches . . . 29
4.3.1. Bounding the function descent . . . 30
4.3.2. Armijo backtracking . . . 30
4.3.3. Wolfe conditions . . . 31

4.4. Gradient descent: conditioning and rates . . . 32
4.5. Nonlinear conjugate gradients . . . 33
4.6. Newton's method . . . 33
4.6.1. Local quadratic convergence . . . 34
4.6.2. Modified and truncated Newton methods . . . 35
4.7. Quasi-Newton methods . . . 36
4.7.1. BFGS . . . 36
4.7.2. Limited-memory BFGS . . . 36
4.8. The structure of nonlinear least squares . . . 37
4.8.1. The linear case and residual orthogonality . . . 37
4.8.2. Gradient and Hessian: a derivation . . . 37
4.8.3. Weighted least squares . . . 38
4.9. Gauss–Newton: derivation, geometry and convergence . . . 38
4.9.1. Two derivations of the same step . . . 38
4.9.2. Characterization of the local behavior for Gauss-Newton . . . 39
4.10. Numerical linear algebra and scaling . . . 41
4.11. Trust-region methods . . . 42
4.12. Levenberg–Marquardt . . . 43
4.12.1. A regularized linearized problem . . . 43
4.12.2. The trust-region subproblem and its multiplier . . . 43
4.12.3. Model agreement and acceptance . . . 44
4.12.4. The relation between damping and the local convergence rate 45
4.13. Implementation and interpretation . . . 45
4.13.1. Stopping criteria and derivative checks . . . 46
4.14. Which method should we choose? . . . 46

5. Discrete mathematics and data structures 47
5.1. Graphs and their representation . . . 47
5.2. Trees, queues, stacks, and heaps . . . 47
5.3. Breadth-first search . . . 48
5.4. Depth-first search . . . 49
5.5. Dijkstra's algorithm . . . 49

II. Case studies 51

6. Population models and epidemics 53
6.1. From balance laws to population models . . . 53
6.2. The SIR model . . . 55
6.2.1. A first integral and the final size . . . 59
6.3. Adding detection and treatment: an SIRT model . . . 60
6.4. Tracking exposure before infectiousness . . . 60
6.4.1. Other useful variations . . . 61

6.5. Implementing the forward ODE solver . . . 62
6.5.1. A vector-valued implementation . . . 62
6.5.2. Choosing the integrator . . . 64
6.5.3. Two simplified numerical experiments . . . 65
6.6. Parameter estimation as an inverse problem . . . 66
6.7. COVID-19 in South Korea . . . 67
6.7.1. Data and observation model . . . 67
6.7.2. Forward solver and nonlinear least squares . . . 68
6.7.3. Verification on future data . . . 71
6.7.4. Extending this idea to a more general workflow . . . 72

7. Computed Tomography **75**
7.1. The physical process . . . 75
7.2. The mathematical forward model . . . 76
7.3. Inverse problem setup . . . 77
7.4. Numerical solution of the (inverse) problem . . . 77
7.4.1. Discretization . . . 77
7.4.2. Solution of the discrete problem and regularization . . . 81

8. Optimal heating **85**
8.1. A floor plan as computational geometry . . . 85
8.2. Finite-difference discretization . . . 86
8.2.1. Why discretize fluxes? . . . 87
8.2.2. Steady state . . . 88
8.2.3. Time integration by implicit Euler . . . 89
8.3. Setting up a PDE-constrained optimization problem . . . 89
8.3.1. Differentiating through the equilibrium equation . . . 90
8.3.2. The optimization problem in practice . . . 91
8.4. Time-dependent problems . . . 92
8.4.1. From the parabolic PDE to a vector ODE . . . 92
8.4.2. The spectrum and stiffness . . . 93
8.4.3. Implicit integrators for the stiff system . . . 94

9. Road networks and graph models **97**
9.1. Graphs: mathematical foundations . . . 97
9.1.1. Graphs, weights, and representations . . . 97
9.1.2. Walks, paths, and distances . . . 98
9.1.3. Bipartite graphs . . . 100
9.1.4. The graph Laplacian: connectivity, cuts, and spectrum . . . 101
9.2. Centrality measures . . . 105
9.2.1. Centrality as a modelling construction . . . 105
9.2.2. Local centrality and accessibility . . . 106
9.2.3. Shortest-path betweenness . . . 107
9.2.4. Eigenvector and Katz centrality . . . 109

9.2.5. Matrix functions: communicability and closed walks 110
9.2.6. Hubs, authorities, and HITS 110
9.2.7. Random walks and PageRank 112
9.2.8. Electrical flows and effective resistance 113
9.2.9. Choosing a centrality measure 114
9.3. Diffusion on graphs 114
9.3.1. From edge exchanges to a balance law 115
9.3.2. Conservation, positivity, and equilibration 116
9.3.3. Spectral modes and the time scale of diffusion 117
9.3.4. Time discretization and sparse computation 118
9.3.5. A numerical experiment and validation checks 119
9.4. Resilience and network interventions 120
9.4.1. Performance before and after a closure 120
9.4.2. Structural baselines and sensitivity 121
9.5. Numerical experiments: the driving network of Pisa 121
9.5.1. Acquire and inspect the network: notebooks 00–01 121
9.5.2. Accessibility and shortest routes: notebooks 02–03 122
9.5.3. Recursive scores and electrical alternatives: notebooks 04–05 122
9.5.4. Closure validation and streets as vertices: notebooks 05–06 123

10. Image classification and low-rank adaptation 125
10.1. The supervised classification problem 125
10.1.1. Training, validation, and test observations 126
10.1.2. Preprocessing and augmentation 126
10.2. From logits to a loss function 127
10.3. Affine and nonlinear classifiers 128
10.4. Mini-batches and backpropagation 129
10.5. Convolutional networks: using image geometry 130
10.5.1. Equivariance, pooling, and receptive fields 131
10.6. Transfer learning and frozen representations 132
10.7. Vision Transformers 133
10.7.1. Patches and positional information 133
10.7.2. Scaled dot-product self-attention 133
10.7.3. Transformer blocks 134
10.8. Low-rank adaptation 134
10.8.1. Initialization and gradients 135
10.8.2. LoRA inside self-attention 136
10.9. Choosing an adaptation strategy 136
10.10. Evaluation beyond accuracy 138
10.10.1. Confidence and calibration 138
10.11. From the theory to the notebooks 139
10.12. Summary and exercises 140

11. Structural health through vibrations **141**
11.1. Elasticity theory 141

Topics and course structure

These are the lecture notes for the course Numerical Simulation of Mathematical Models. This course is aimed at master students looking for some practical experience in dealing with the numerical treatment of models coming from applications.

The course targets math students, and focuses on the interactions that a mathematician has when dealing with practitioners such as engineers, physicists, medical doctors, and many others. The course is structured as follows:

Modeling tools The first part gives a high-level overview of the mathematical tools often found in this context. These include: finite elements, optimization, eigenvalues and stability, integration of ODEs, and graph algorithms and data structures. The treatment of these topics is necessarily brief and concise. It is meant as a quick recap for those that are already familiar with these topics, and as a quick overview with pointers to gain a deeper understanding for the others. A good grasp of the main general concepts behind numerical analysis is assumed throughout.

Case studies A few selected case studies are considered. They are chosen as a test bed to better understand how the considered tools are used in practice, and the subtleties in the process. These notes will possibly include more case studies than the ones that could be feasible to fully cover in a semester, accumulated throughout the years in which this course is taught.

The course lectures will not necessarily follow the linear structure of these notes. For instance, the modeling tools will be introduced as course unfolds and as they are needed to tackle the problems at hand.

AI policy

LLM coding models can be a helpful tool while coding and experimenting; these lecture note are being written in Fall 2026, at a moment in time when AI can be very effective in writing code, and has in fact been used to generate some of the code for this course. The policy for AI use by students in the course is as follows: you are free to do so in any measure you want, but recall that:

- To actually learn something, it is often useful to spend some time reasoning on it, so we advise using an LLM for possible hints, corrections and so on.
- On the other hand, AI is often useful to accelerate all steps that would take a substantial amount of time but that you know exactly how to carry out. If the

issue is only length and boring boiler plate code, then LLMs can be a useful support.

- Similarly, sometimes you will find that you have to interact with external libraries that need to be called with certain parameters. Reading the documentation is one option (and the best one), but sometimes having the code generated and just checking that is doing the right thing is faster.

For assignments, a reasonable use of AI for the same tasks listed above is fine; just keep in mind that you will have to discuss your work during the exam and in the classroom, and in that context having a good and deep understanding of it is crucial.

Part I.

Modeling tools

1. Solving linear systems and computing eigenvalues

In this chapter we briefly recall the numerical methods for linear systems and eigenvalue problems, assuming familiarity with their basic properties. Our aim is to understand which method to choose in a given application. We will consider the following topics:

- Fixed-point iterations;
- Krylov methods for symmetric and unsymmetric linear systems.
- Least squares: QR and Krylov methods.
- Eigenvalues: small-scale (QR, no details) vs large-scale (Power and subspace iteration, Krylov methods, Jacobi-Davidson).

Linear systems occur at nearly every stage of numerical modeling. They may arise directly from a linear model, from the discretization of a PDE, or as inner problems in nonlinear equations, optimization, and implicit time integration. We write them as

$$A\boldsymbol{x} = \boldsymbol{b}, \qquad A \in \mathbb{R}^{n\times n}.$$

Before choosing a solver, we need to identify the properties of A that we can exploit. In particular, we consider sparsity, symmetry, positive definiteness, conditioning, and whether several systems with the same matrix must be solved.

1.1. Sensitivity, error, and residual

Let us distinguish the error in an approximate solution from the residual that we can compute.

Definition 1.1.1. Given $\widehat{\boldsymbol{x}}$, we define the residual as

$$\boldsymbol{r} = \boldsymbol{b} - A\widehat{\boldsymbol{x}},$$

whereas the error is $\boldsymbol{e} = \boldsymbol{x} - \widehat{\boldsymbol{x}}$. They satisfy $A\boldsymbol{e} = \boldsymbol{r}$.

A small residual is easy to verify, but it guarantees a small error only if A is reasonably well conditioned. In a consistent norm,

$$\frac{\|\boldsymbol{e}\|}{\|\boldsymbol{x}\|} \le \kappa(A)\frac{\|\boldsymbol{r}\|}{\|\boldsymbol{b}\|}, \qquad \kappa(A) = \|A\|\,\|A^{-1}\|. \tag{1.1}$$

We therefore cannot expect a solver to recover information lost through the conditioning of the mathematical problem. Scaling the variables and equations often improves both conditioning and the interpretation of stopping criteria.

In computations it is preferable to monitor a relative backward error, for example

$$\eta = \frac{\|\boldsymbol{b} - A\widehat{\boldsymbol{x}}\|}{\|A\|\,\|\widehat{\boldsymbol{x}}\| + \|\boldsymbol{b}\|},$$

rather than an unscaled residual. This quantity measures how much the data would need to change for the computed vector to be an exact solution.

1.2. Direct methods

We first consider direct methods. Gaussian elimination gives a factorization of A as LU, using permutations for numerical stability. For a dense matrix its cost is $O(n^3)$ operations and $O(n^2)$ memory. If A is symmetric positive definite, the Cholesky factorization $A = LL^T$ is cheaper and preserves symmetry. The QR factorization is more expensive, but has strong stability properties and is the standard direct method for least-squares problems.

The situation is different for sparse matrices. The factors generally contain entries in positions where A was zero, a phenomenon called *fill-in*. Sparse direct solvers reorder the unknowns before factorization to limit this fill. They are attractive for small and medium problems, and especially when the same matrix is used with many right-hand sides: the expensive factorization is performed once, then each new system requires only triangular substitutions. The implicit heat solver in Chapter 8 uses precisely this pattern.

1.3. Stationary iterations

A different approach is to split $A = M - N$, choosing M so that the corresponding linear systems are easy to solve. We then consider the fixed-point iteration

$$M\boldsymbol{x}^{(k+1)} = N\boldsymbol{x}^{(k)} + \boldsymbol{b}$$

has error propagation $\boldsymbol{e}^{(k+1)} = M^{-1}N\boldsymbol{e}^{(k)}$. It converges for every starting vector exactly when $\rho(M^{-1}N) < 1$, where ρ is the spectral radius. Jacobi takes M to be the diagonal of A, while Gauss–Seidel takes its lower triangular part.

These schemes are rarely competitive as stand-alone solvers for a difficult large system. Nevertheless, they are important as smoothers in multigrid and as building blocks for preconditioners. They also show why we need to consider the spectrum of the iteration matrix, in addition to the cost of a single step, when assessing convergence.

1.4. Krylov methods

For large problems, we can instead construct approximations in a sequence of low-dimensional spaces. Starting from $\boldsymbol{x}^{(0)}$, a Krylov method searches for a correction in

$$\mathcal{K}_k(A, \boldsymbol{r}^{(0)}) = \operatorname{span}\{\boldsymbol{r}^{(0)}, A\boldsymbol{r}^{(0)}, \ldots, A^{k-1}\boldsymbol{r}^{(0)}\}.$$

Only products with A are required, so a sparse matrix need not be factorized and may not even need to be stored explicitly.

If the matrix is symmetric positive definite, we usually choose the conjugate-gradient method (CG). It minimizes the energy norm of the error over the current Krylov space and, in exact arithmetic, satisfies the indicative bound

$$\frac{\|\boldsymbol{e}^{(k)}\|_A}{\|\boldsymbol{e}^{(0)}\|_A} \leq 2\left(\frac{\sqrt{\kappa_2(A)}-1}{\sqrt{\kappa_2(A)}+1}\right)^k.$$

MINRES also exploits symmetry but permits an indefinite matrix. For a general non-symmetric matrix, GMRES minimizes the residual norm. Its orthogonal basis grows with the iteration count, so restarted GMRES is often used to control memory. BiCGSTAB uses short recurrences and less memory, at the price of less predictable convergence.

1.5. Preconditioning

For large problems, the preconditioner is often more important than the choice between Krylov variants. We replace the original system by one involving an easily invertible matrix $P \approx A$, such as

$$P^{-1}A\boldsymbol{x} = P^{-1}\boldsymbol{b}.$$

We seek to cluster the spectrum, or otherwise improve the properties of the preconditioned operator, while keeping applications of P^{-1} inexpensive. Common choices include diagonal scaling, incomplete LU or Cholesky factorizations, domain decomposition, and multigrid. A valid preconditioner for CG must preserve symmetry and positive definiteness. To compare preconditioners, we need to consider total time and memory, including setup costs. The number of iterations alone does not provide a complete comparison.

1.6. Least-squares problems

For an overdetermined model $A\boldsymbol{x} \simeq \boldsymbol{b}$, the least-squares solution minimizes $\|A\boldsymbol{x} - \boldsymbol{b}\|_2$. The normal equations

$$A^TA\boldsymbol{x} = A^T\boldsymbol{b}$$

give a square system. However, we have $\kappa_2(A^TA) = \kappa_2(A)^2$. A QR factorization is therefore the usual dense direct method, while the singular-value decomposition is preferred when rank deficiency or very poor conditioning must be diagnosed. For large sparse problems, LSQR and LSMR construct Krylov approximations without explicitly forming A^TA.

1.7. A practical choice

For dense or moderately sized sparse problems, a reliable direct factorization is a suitable first choice. For large sparse symmetric positive-definite problems, we can start with preconditioned CG. MINRES is appropriate for symmetric indefinite systems, whereas a preconditioned GMRES-type method can be used for nonsymmetric systems. When a matrix is used repeatedly, we can reuse its factorization or preconditioner. In all cases, we check a scaled residual and verify that tighter tolerances or a refined model do not materially change the result.

1.8. The singular value decomposition (SVD)

We conclude this recap with the singular value decomposition (SVD), which is available for every matrix.

Definition 1.8.1. Let $A \in \mathbb{R}^{m\times n}$ and assume, for simplicity, that $m \geq n$. Then the SVD of A takes the form

$$\begin{aligned} A &= U\Sigma V^T \\ &= \textstyle\sum_{i=1}^{n} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T \end{aligned} \text{ , where } \begin{aligned} \Sigma &= \operatorname{diag}(\sigma_1, \ldots, \sigma_n) \in \mathbb{R}^{n\times n}, \\ &\quad \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0, \\ U &= [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n] \in \mathbb{R}^{m\times n} \text{ has orthonormal columns,} \\ V &= [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n] \in \mathbb{R}^{n\times n} \text{ is orthogonal.} \end{aligned}$$

The (diagonal) entries of Σ are the singular values of A; the (orthonormal) columns of U and V are the left and right singular vectors of A, respectively.

Once the SVD of A is available, we can recover several useful properties of A. In particular, we have:

- $r = \operatorname{rank}(A)$ is the number of nonzero singular values of A, i.e.,

$$\sigma_r > 0, \quad \sigma_{r+1}(= \ldots \sigma_n) = 0\,.$$

- $\operatorname{range}(A) = \operatorname{span}\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r\}$, $\operatorname{null}(A) = \operatorname{span}\{\boldsymbol{v}_{r+1}, \ldots, \boldsymbol{v}_n\}$
- We have the following relations:

$$A\boldsymbol{v}_i = \sigma_i \boldsymbol{u}_i, \quad , \|A\boldsymbol{v}_i\|_2 = \sigma_i, \quad i = 1, \ldots, n\,.$$

- $\kappa_2(A)$, the condition number of A with respect to the spectral norm, is equal to the ratio σ_1/σ_n.

2. Integration of ordinary differential equations

We consider an initial-value problem

$$\boldsymbol{y}'(t) = \boldsymbol{f}(t, \boldsymbol{y}(t)), \qquad \boldsymbol{y}(t_0) = \boldsymbol{y}_0,$$

where $\boldsymbol{y} : I \to \mathbb{R}^m$. Such systems may describe a lumped physical model or arise after spatial discretization of a PDE, as in the method of lines. We seek approximations $\boldsymbol{y}_n \simeq \boldsymbol{y}(t_n)$, taking into account accuracy, stability, and computational cost.

2.1. Consistency and convergence

We first consider one-step methods, which can be written in the form

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \Delta t\, \Phi(t_n, \boldsymbol{y}_n, \Delta t).$$

Definition 2.1.1. The local truncation error is the defect obtained by inserting the exact solution into one step. A method has order p if this defect is $O(\Delta t^{p+1})$.

Over a fixed time interval, stable propagation usually gives a global error $O(\Delta t^p)$. Order describes the asymptotic behavior, but the error constant and the stability restriction determine which method is useful in practice.

The explicit and implicit Euler methods are, respectively,

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \Delta t\, \boldsymbol{f}(t_n, \boldsymbol{y}_n),$$

and

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \Delta t\, \boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1}).$$

Both have order one. Explicit Euler evaluates $\boldsymbol{f}$ once; implicit Euler requires the solution of a generally nonlinear equation at every step.

2.2. Absolute stability and stiffness

To study stability, we apply the method to the test equation $y' = \lambda y$. A method produces $y_{n+1} = R(z)y_n$, where $z = \Delta t\lambda$ and R is its stability function. The absolute stability region is

$$\mathcal{S} = \{z \in \mathbb{C} : |R(z)| \le 1\}.$$

Explicit Euler has $R(z) = 1 + z$ and is stable only inside the disk $|1 + z| \leq 1$. Implicit Euler has $R(z) = (1-z)^{-1}$ and its region contains the entire left half-plane. It is both A-stable and L-stable: rapidly decaying modes are strongly damped even for large negative z.

We call a problem stiff when an explicit method must take steps much smaller than those required to resolve the behaviour of interest in order to remain stable. Diffusion discretizations are a standard example: their most negative eigenvalues grow in magnitude like h^{-2}, leading to an explicit stability restriction $\Delta t = O(h^2)$. An implicit method trades this restriction for linear or nonlinear solves.

2.2.1. A simple stiff problem

Example 2.2.1. Let us consider the linear initial-value problem

$$\boldsymbol{y}'(t) = A\boldsymbol{y}(t), \qquad A = \begin{bmatrix} -1 & 0 \\ 0 & -10^{20} \end{bmatrix}, \qquad \boldsymbol{y}(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Its exact solution is simply

$$\boldsymbol{y}(t) = \begin{bmatrix} e^{-t} \\ 0 \end{bmatrix}.$$

Thus the solution of interest is smooth and evolves on a time scale of order one. The eigenvalue -10^{20} does not contribute to this exact solution. Nevertheless, a reliable numerical method must not amplify small perturbations in its second component, which may be introduced by roundoff, imperfect initial data, or coupling to other parts of a larger model. Note that even if the second component is non-zero, it is immediately damped to zero by the large negative eigenvalue 10^{20}; hence, we can neglect it when studying the continuous dynamical system. However, it becomes relevant for the discrete system if we use an explicit integrator.

For explicit Euler the iteration matrix is

$$I + \Delta t A = \begin{bmatrix} 1 - \Delta t & 0 \\ 0 & 1 - 10^{20}\Delta t \end{bmatrix}.$$

Stability of the second component requires

$$|1 - 10^{20}\Delta t| \leq 1, \qquad \text{hence} \qquad \Delta t \leq 2 \cdot 10^{-20}.$$

If its second component were represented as exactly zero and arithmetic were exact, explicit Euler would preserve that invariant subspace and the restriction would not be observed. In computation, however, an arbitrarily small perturbation grows catastrophically when the bound is violated. To integrate up to $T = 1$ robustly, explicit Euler therefore needs at least $5 \cdot 10^{19}$ steps, even though roughly 10^2 steps might already resolve the visible component to a modest accuracy.

Implicit Euler instead gives

$$\boldsymbol{y}^{n+1} = (I - \Delta t A)^{-1}\boldsymbol{y}^n, \qquad (I - \Delta t A)^{-1} = \begin{bmatrix} (1 + \Delta t)^{-1} & 0 \\ 0 & (1 + 10^{20}\Delta t)^{-1} \end{bmatrix}.$$

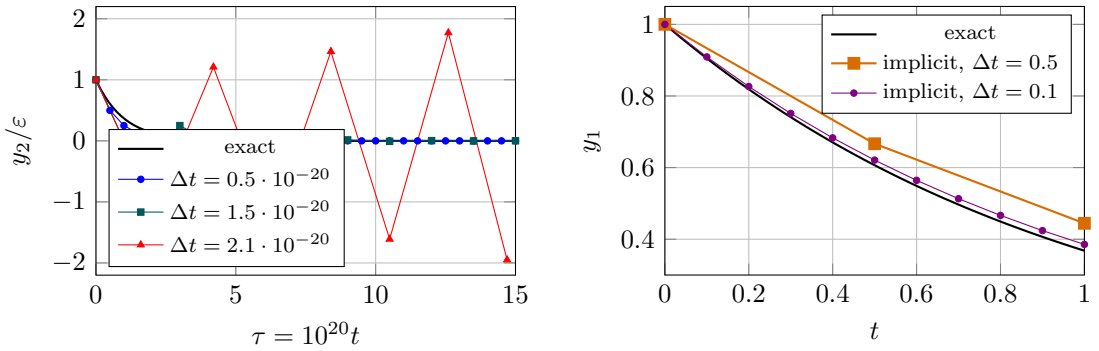


Figure 2.1.: Euler discretizations of the stiff problem. Left: explicit Euler applied to a perturbation in the fast component; the largest step lies outside the stability interval. Right: implicit Euler applied to the slow component with two accuracy-driven step sizes.

Both amplification factors have magnitude below one for every $\Delta t > 0$. The step size may consequently be selected to resolve the slow mode and meet the desired accuracy, rather than to stabilize an irrelevant fast mode. For example, $\Delta t = 10^{-2}$ requires only 100 steps on $[0, 1]$.

In this diagonal two-dimensional example, both an explicit step and an implicit step cost only a few scalar operations, so the difference in the number of steps makes implicit Euler much cheaper. For a general system, an explicit step usually requires only a matrix–vector product, whereas an implicit step requires solving a linear system with $I - \Delta t A$. The latter is more expensive per step, but if A and Δt are fixed its factorization can be computed once and reused. Stiffness makes this additional work worthwhile whenever the stability-limited explicit step is far smaller than the step required for accuracy.

Figure 2.1 illustrates both effects. In the left panel, a small perturbation $y_2(0) = \varepsilon$ is introduced and the normalized fast component is plotted against the scaled time $\tau = 10^{20}t$. With $\Delta t = 0.5 \cdot 10^{-20}$ it decays monotonically, while $\Delta t = 1.5 \cdot 10^{-20}$ produces harmless decaying oscillations. Increasing the step only slightly beyond the stability boundary, to $\Delta t = 2.1 \cdot 10^{-20}$, makes the perturbation grow. The right panel instead follows the slow component with implicit Euler: both displayed step sizes are stable, and their difference concerns accuracy alone.

2.3. Runge–Kutta methods

We can obtain higher-order one-step methods by introducing internal stages. A Runge–Kutta method computes

$$\boldsymbol{Y}_i = \boldsymbol{y}_n + \Delta t \sum_{j=1}^{s} a_{ij} \boldsymbol{f}(t_n + c_j \Delta t, \boldsymbol{Y}_j),$$

and then sets

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \Delta t \sum_{i=1}^{s} b_i \boldsymbol{f}(t_n + c_i \Delta t, \boldsymbol{Y}_i).$$

If the coefficient matrix (a_{ij}) is strictly lower triangular, the method is explicit. Classical RK4 is a robust fourth-order choice for nonstiff problems. Embedded pairs, such as methods of orders four and five sharing the same stages, estimate the local error and adapt the step size automatically.

Implicit Runge–Kutta methods can have excellent stability but couple all stages in one nonlinear system. Diagonally implicit Runge–Kutta (DIRK) methods use a lower triangular coefficient matrix and solve the stages sequentially. They are useful when stiffness requires implicit stability but a fully coupled solve would be too expensive.

2.4. Linear multistep methods

An alternative is to use several previous solution values. A linear multistep method has the form

$$\sum_{j=0}^{k} \alpha_j \boldsymbol{y}_{n+j} = \Delta t \sum_{j=0}^{k} \beta_j \boldsymbol{f}(t_{n+j}, \boldsymbol{y}_{n+j}).$$

Adams–Bashforth methods are explicit, Adams–Moulton methods are implicit, and backward differentiation formulae (BDF) are designed for stiff problems. Once started, a multistep scheme may need only one new function evaluation per step, but changing the step size and order is more involved than for a one-step method. It also needs starting values, normally supplied by a Runge–Kutta method. Zero-stability, together with consistency, is required for convergence.

2.5. Solving implicit steps

Let us examine the equations that arise at an implicit step. For implicit Euler, we define

$$\boldsymbol{G}(\boldsymbol{z}) = \boldsymbol{z} - \boldsymbol{y}_n - \Delta t\, \boldsymbol{f}(t_{n+1}, \boldsymbol{z}).$$

Newton's method solves $\boldsymbol{G}(\boldsymbol{z}) = 0$ using systems with matrix $I - \Delta t\, \partial \boldsymbol{f}/\partial \boldsymbol{y}$. Modified Newton methods reuse a Jacobian or its factorization across iterations and steps. For a linear semidiscrete PDE, the nonlinear iteration disappears and the same system matrix can often be reused throughout the integration.

We need to choose a nonlinear tolerance tighter than the requested time integration accuracy. Solving every step to machine precision, however, would introduce unnecessary work. A failed nonlinear iteration often signals that the step is too large or that the Jacobian or preconditioner is inadequate.

2.6. Error control and method selection

Adaptive solvers estimate a local error e_n and compare a scaled norm against absolute and relative tolerances. A typical controller updates the step approximately as

$$\Delta t_{\text{new}} = s\,\Delta t \left(\frac{1}{e_n}\right)^{1/(p+1)},$$

with a safety factor $s < 1$ and bounds on sudden changes. Different components need suitable scales; otherwise a large physical variable can hide an unacceptable error in a small one.

For a nonstiff problem, an explicit adaptive Runge–Kutta method is a suitable choice. If stability limits the explicit step more severely than accuracy, we can use BDF, DIRK, or another stiff solver. We may also need to preserve qualitative properties such as positivity, invariants, or energy decay; these can require a specialized method. Finally, we verify the computation by tightening tolerances and checking that the quantities of interest change as predicted.

3. Computing eigenvalues

We now consider the computation of eigenvalues. Let us first recall the definition of an eigenpair. An eigenpair of $A \in \mathbb{C}^{n\times n}$ satisfies

$$A\boldsymbol{v} = \lambda\boldsymbol{v}, \qquad \boldsymbol{v} \neq 0.$$

Eigenvalues determine vibration frequencies, growth and decay rates, stability, and the convergence of iterations. In many applications we only need part of the spectrum: a structural model may need only its lowest frequencies, while a stability analysis may seek the eigenvalues with largest real part.

3.1. Sensitivity and residuals

For an approximate eigenpair $(\widehat{\lambda}, \widehat{\boldsymbol{v}})$ with $\|\widehat{\boldsymbol{v}}\|_2 = 1$, the computable residual is

$$\boldsymbol{r} = A\widehat{\boldsymbol{v}} - \widehat{\lambda}\widehat{\boldsymbol{v}}.$$

For a Hermitian matrix, an eigenvalue lies within $\|\boldsymbol{r}\|_2$ of $\widehat{\lambda}$. For a nonnormal matrix, small perturbations can move eigenvalues much farther; eigenvector conditioning and pseudospectra may then be more informative than eigenvalues alone. Repeated or tightly clustered eigenvalues also call for computing an invariant subspace rather than interpreting individual eigenvectors.

3.2. Dense problems: the QR algorithm

When most or all eigenvalues of a dense matrix are required, the standard approach first reduces A by orthogonal similarity transformations to Hessenberg form, or to tridiagonal form when A is Hermitian. Shifted QR iterations then converge to a Schur form

$$A = QTQ^*,$$

where Q is unitary and T is upper triangular. The eigenvalues are the diagonal entries of T. The reduction costs $O(n^3)$ operations, so this route is appropriate when n is moderate and a substantial part of the spectrum is desired. In practice, we use a tested library routine instead of implementing QR iteration directly.

3.3. Power and inverse iteration

A simple way to approximate a dominant eigenvector is to apply the matrix repeatedly. This gives the power iteration,

$$\boldsymbol{w}^{(k+1)} = A\boldsymbol{v}^{(k)}, \qquad \boldsymbol{v}^{(k+1)} = \frac{\boldsymbol{w}^{(k+1)}}{\|\boldsymbol{w}^{(k+1)}\|}.$$

Under standard assumptions it converges to an eigenvector associated with the eigenvalue of largest magnitude, at a rate governed by $|\lambda_2/\lambda_1|$. The Rayleigh quotient

$$\widehat{\lambda} = \frac{\boldsymbol{v}^* A \boldsymbol{v}}{\boldsymbol{v}^* \boldsymbol{v}}$$

provides the eigenvalue estimate.

To find an eigenvalue near a target σ, we can instead apply power iteration to $(A - \sigma I)^{-1}$. Shift-and-invert iteration requires a linear solve at every step, but a fixed factorization of $A - \sigma I$ can be reused. Eigenvalues nearest σ are transformed into dominant ones, which often makes convergence rapid. Rayleigh-quotient iteration updates the shift and can converge cubically for Hermitian matrices near a simple eigenpair, although its changing shifts require new linear systems.

Block or subspace iteration propagates several vectors at once and orthonormalizes them after each application. It is more robust for clusters and returns an approximation to the whole desired invariant subspace.

3.4. Krylov eigensolvers

For a large sparse matrix, Arnoldi iteration constructs an orthonormal basis V_k of $\mathcal{K}_k(A, \boldsymbol{v})$ and the relation

$$AV_k = V_k H_k + h_{k+1,k}\boldsymbol{v}_{k+1}\boldsymbol{e}_k^T,$$

where $H_k = V_k^* A V_k$ is a small Hessenberg matrix. Its eigenpairs, called Ritz pairs, approximate eigenpairs of A. For a Hermitian matrix, Arnoldi reduces to the Lanczos process and H_k is tridiagonal, yielding short recurrences and lower storage.

As with GMRES, an unrestarted Arnoldi basis grows. Implicitly restarted Arnoldi and thick-restart Lanczos retain useful spectral information while limiting memory. Extremal eigenvalues are usually found first; interior eigenvalues are commonly exposed with a shift-and-invert transformation. We need to include the resulting linear solves in the cost of the eigensolver and choose their preconditioners accordingly.

3.5. Jacobi–Davidson

Jacobi–Davidson targets an eigenvalue while expanding a search subspace with an approximate solution of the correction equation

$$(I - \boldsymbol{v}v^*)(A - \theta I)(I - \boldsymbol{v}v^*)\boldsymbol{s} = -\boldsymbol{r}, \qquad \boldsymbol{s} \perp \boldsymbol{v},$$

where $(\theta, \boldsymbol{v})$ is the current Ritz pair. The correction equation need not be solved accurately; a few preconditioned Krylov steps may suffice. This flexibility makes Jacobi–Davidson attractive for difficult interior eigenvalues when an exact shift-and-invert factorization is too costly.

3.6. Generalized eigenvalue problems

When discretizing a PDE, we frequently obtain a generalized eigenvalue problem of the form

$$K\boldsymbol{v} = \lambda M\boldsymbol{v},$$

where K is a stiffness matrix and M is a mass matrix. If K is symmetric and M is symmetric positive definite, eigenvectors can be chosen M-orthonormal. Forming $M^{-1}K$ explicitly destroys symmetry and sparsity; generalized symmetric eigensolvers instead work with the pair (K, M) or with factorizations that preserve its structure.

For a small dense problem, we can compute a Schur or generalized Schur decomposition. For a large sparse problem, we first specify which part of the spectrum is needed. We then use Lanczos for Hermitian problems or Arnoldi for general matrices, with shift-and-invert when appropriate. Block methods are useful for clusters, while Jacobi–Davidson allows approximate inner solves when accurate ones would be too expensive. In every case, we report eigenpair residuals and check that the computed eigenvalues are the ones relevant to the physical question.

4. Optimization

Optimization arises in mathematical modelling when we need to choose parameters, controls, or designs to minimize a quantitative objective. In this chapter we consider smooth unconstrained problems. We shall derive algorithms from local models, distinguishing existence of a solution, global stationarity, and local convergence. The same methods also provide components for constrained algorithms.

4.1. Formulation, notation, and existence

We study

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}). \tag{4.1}$$

The word *continuous* refers to the decision variables: they range over a continuum rather than a discrete set. We concentrate on smooth objectives. Constraints require additional ideas, such as tangent cones, Lagrange multipliers and constraint qualifications, which are outside the scope of this chapter.

Throughout, vectors are columns, $\langle \boldsymbol{x}, \boldsymbol{y}\rangle = \boldsymbol{x}^T\boldsymbol{y}$ and $\|\boldsymbol{x}\|$ is the Euclidean norm. For matrices, $\|A\|$ is its induced operator norm. A symmetric matrix A satisfies $A \succeq 0$ if $\boldsymbol{v}^T A\boldsymbol{v} \geq 0$ for every $\boldsymbol{v}$, and $A \succ 0$ if the inequality is strict for every nonzero $\boldsymbol{v}$. Derivatives of vector-valued maps act on column increments. All local statements take place in an open neighbourhood on which the indicated derivatives exist.

Definition 4.1.1. A point $\boldsymbol{x}_*$ is a *local minimizer* if some $\delta > 0$ satisfies $f(\boldsymbol{x}_*) \leq f(\boldsymbol{x})$ whenever $\|\boldsymbol{x} - \boldsymbol{x}_*\| < \delta$. It is a *strict local minimizer* if the inequality is strict for $\boldsymbol{x} \neq \boldsymbol{x}_*$. It is a *global minimizer* if $f(\boldsymbol{x}_*) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathbb{R}^n$. A differentiable objective has a *stationary point* at $\boldsymbol{x}_*$ if $\nabla f(\boldsymbol{x}_*) = 0$.

Before considering algorithms, we need to distinguish existence of a solution from its characterization and computation. For example, $f(x) = e^x$ is positive and smooth on $\mathbb{R}$ but does not attain its infimum.

Proposition 4.1.2 (A sufficient condition for existence)**.** *If $f : \mathbb{R}^n \to \mathbb{R}$ is continuous and coercive, meaning $f(\boldsymbol{x}) \to +\infty$ as $\|\boldsymbol{x}\| \to \infty$, then f attains a global minimum.*

Proof. Let us choose any $\boldsymbol{x}_0$ and consider the sublevel set $K = \{\boldsymbol{x} : f(\boldsymbol{x}) \leq f(\boldsymbol{x}_0)\}$ is nonempty and closed by continuity. Coercivity makes it bounded. In finite dimensions it is compact, so Weierstrass's theorem gives a minimum on K. Outside K the value exceeds $f(\boldsymbol{x}_0)$, hence that minimum is global. □

4.2. First- and second-order optimality

Theorem 4.2.1 (Necessary conditions)**.** *Let $\boldsymbol{x}_*$ be an unconstrained local minimizer.*

1. *If f is differentiable at $\boldsymbol{x}_*$, then $\nabla f(\boldsymbol{x}_*) = 0$.*
2. *If f is C^2 near $\boldsymbol{x}_*$, then $\nabla^2 f(\boldsymbol{x}_*) \succeq 0$.*

Proof. For any $\boldsymbol{d} \in \mathbb{R}^n$, we define $\phi(t) = f(\boldsymbol{x}_* + t\boldsymbol{d})$. The point 0 is a local minimizer of ϕ. Therefore $\phi'(0) = \nabla f(\boldsymbol{x}_*)^T \boldsymbol{d} = 0$. Taking $\boldsymbol{d} = \nabla f(\boldsymbol{x}_*)$ proves (i). For (ii), scalar second-order necessity gives $\phi''(0) = \boldsymbol{d}^T \nabla^2 f(\boldsymbol{x}_*)\boldsymbol{d} \geq 0$ for every $\boldsymbol{d}$. □

Theorem 4.2.2 (Second-order sufficiency)**.** *Suppose $f \in C^2$ near $\boldsymbol{x}_*$, $\nabla f(\boldsymbol{x}_*) = 0$ and $\mathcal{H}_* := \nabla^2 f(\boldsymbol{x}_*) \succ 0$. Then there exist $c, \delta > 0$ such that*

$$f(\boldsymbol{x}_* + \boldsymbol{p}) \geq f(\boldsymbol{x}_*) + c\,\|\boldsymbol{p}\|^2 \qquad (\|\boldsymbol{p}\| < \delta).$$

In particular, $\boldsymbol{x}_*$ is a strict local minimizer.

Proof. Let $\mu = \lambda_{\min}(\mathcal{H}_*) > 0$. Taylor expansion gives

$$f(\boldsymbol{x}_* + \boldsymbol{p}) - f(\boldsymbol{x}_*) = \tfrac{1}{2}\boldsymbol{p}^T \mathcal{H}_* \boldsymbol{p} + o(\|\boldsymbol{p}\|^2).$$

For sufficiently small $\boldsymbol{p}$, the magnitude of the remainder is at most $\mu\,\|\boldsymbol{p}\|^2/4$. The claim follows with $c = \mu/4$. □

Remark 4.2.3. Positive semidefiniteness alone does not allow us to distinguish a minimum from a maximum. Both $f(x) = x^4$ and $f(x) = -x^4$ satisfy $f'(0) = f''(0) = 0$, but one has a strict minimum and the other a strict maximum. A strict minimum need not have a positive definite Hessian.

4.2.1. Convexity and global optimality

Definition 4.2.4. On a convex set D, a function is convex if

$$f((1-t)\boldsymbol{x} + t\boldsymbol{y}) \leq (1-t)f(\boldsymbol{x}) + tf(\boldsymbol{y}), \qquad \boldsymbol{x}, \boldsymbol{y} \in D, \quad 0 \leq t \leq 1.$$

It is strictly convex if the inequality is strict for $\boldsymbol{x} \neq \boldsymbol{y}$ and $0 < t < 1$. For a differentiable function on an open convex set, we use the equivalent definition of μ-strong convexity

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x}) + \frac{\mu}{2}\,\|\boldsymbol{y} - \boldsymbol{x}\|^2, \qquad \mu > 0. \tag{4.2}$$

Proposition 4.2.5 (Differential characterizations)**.** *Let $D \subset \mathbb{R}^n$ be open and convex.*

1. *A differentiable f is convex if and only if*

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x}) \qquad (\boldsymbol{x}, \boldsymbol{y} \in D). \tag{4.3}$$

2. *If $f \in C^2(D)$, it is convex if and only if $\nabla^2 f(\boldsymbol{x}) \succeq 0$ for all $\boldsymbol{x} \in D$.*

3. *A stationary point of a differentiable convex function is a global minimizer. A strictly convex function has at most one minimizer.*

Proof. If f is convex, we apply its defining inequality to $\boldsymbol{x} + t(\boldsymbol{y} - \boldsymbol{x})$, subtract $f(\boldsymbol{x})$, divide by $t > 0$, and pass to $t \downarrow 0$. This gives (4.3). Conversely, we set $\boldsymbol{z} = (1-t)\boldsymbol{x} + t\boldsymbol{y}$ and apply (4.3) at $\boldsymbol{z}$ to both endpoints, and combine with weights $1-t, t$. The gradient terms cancel.

For (ii), if f is convex, its restriction $\phi(t) = f(\boldsymbol{x} + t\boldsymbol{d})$ is convex on every admissible line interval, so $\phi''(0) = \boldsymbol{d}^T \nabla^2 f(\boldsymbol{x}) \boldsymbol{d} \geq 0$. Conversely, a nonnegative Hessian makes each line restriction have nonnegative second derivative; scalar convexity on every segment proves convexity. A stationary point satisfies (4.3) with zero linear term. Finally, two distinct minimizers would violate strict convexity at their midpoint. □

We remark that strict convexity does not imply existence of a minimizer: e^x is strictly convex on $\mathbb{R}$. In the C^2 setting, μ-strong convexity is equivalent to $\nabla^2 f(x) \succeq \mu I$ throughout D, by the same line-restriction argument with $f(x) - \mu \|x\|^2 / 2$.

4.3. Local models, descent directions, and line searches

We shall construct methods of the form

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k, \qquad \alpha_k > 0. \tag{4.4}$$

We choose a direction $\boldsymbol{p}_k$ and a step length α_k. We write $\boldsymbol{g}_k = \nabla f(\boldsymbol{x}_k)$ and, when working at a generic point, $\boldsymbol{g}(\boldsymbol{x}) = \nabla f(\boldsymbol{x})$. The local quadratic model is

$$m_k(\boldsymbol{p}) = f(\boldsymbol{x}_k) + \boldsymbol{g}_k^T \boldsymbol{p} + \tfrac{1}{2} \boldsymbol{p}^T B_k \boldsymbol{p}.$$

Here B_k is the Hessian or a symmetric approximation. First-order methods use gradients; Newton and quasi-Newton methods also exploit curvature.

Lemma 4.3.1 (Descent direction)**.** *If f is differentiable at $\boldsymbol{x}$ and $\boldsymbol{g}(\boldsymbol{x})^T \boldsymbol{p} < 0$, then $f(\boldsymbol{x} + \alpha \boldsymbol{p}) < f(\boldsymbol{x})$ for all sufficiently small $\alpha > 0$.*

Proof. Differentiability gives $f(\boldsymbol{x} + \alpha \boldsymbol{p}) = f(\boldsymbol{x}) + \alpha \boldsymbol{g}(\boldsymbol{x})^T \boldsymbol{p} + o(\alpha)$. Dividing by α and using the strict negativity of the limit proves the claim. □

For $\boldsymbol{g} \neq 0$, Cauchy–Schwarz gives

$$\min_{\|\boldsymbol{p}\|=1} \boldsymbol{g}^T \boldsymbol{p} = -\|\boldsymbol{g}\|, \qquad \boldsymbol{p} = -\boldsymbol{g} / \|\boldsymbol{g}\|.$$

Thus steepest descent in the Euclidean norm motivates $\boldsymbol{p} = -\boldsymbol{g}$. For a metric $\|\boldsymbol{p}\|_M = (\boldsymbol{p}^T M \boldsymbol{p})^{1/2}$ with $M \succ 0$, the normalized steepest direction is

$$-\frac{M^{-1}\boldsymbol{g}}{\sqrt{\boldsymbol{g}^T M^{-1} \boldsymbol{g}}}.$$

The unnormalized direction $-M^{-1}\boldsymbol{g}$ shows explicitly why variable scaling changes the method.

4.3.1. Bounding the function descent

Lemma 4.3.2 (Descent lemma)**.** *Suppose $f \in C^1$ and its gradient is L-Lipschitz on the segment from $\boldsymbol{x}$ to $\boldsymbol{x}+\boldsymbol{p}$: $\|\boldsymbol{g}(\boldsymbol{u}) - \boldsymbol{g}(\boldsymbol{v})\| \le L\|\boldsymbol{u}-\boldsymbol{v}\|$. Then*

$$f(\boldsymbol{x}+\boldsymbol{p}) \le f(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})^T\boldsymbol{p} + \frac{L}{2}\|\boldsymbol{p}\|^2. \tag{4.5}$$

Proof. The fundamental theorem of calculus gives

$$f(\boldsymbol{x}+\boldsymbol{p}) - f(\boldsymbol{x}) - \boldsymbol{g}(\boldsymbol{x})^T\boldsymbol{p} = \int_0^1 [\boldsymbol{g}(\boldsymbol{x}+t\boldsymbol{p}) - \boldsymbol{g}(\boldsymbol{x})]^T\boldsymbol{p}\,dt.$$

We bound the integrand from above by $Lt\|\boldsymbol{p}\|^2$ and integrate to obtain the result. □

4.3.2. Armijo backtracking

We choose $c_1, \beta \in (0,1)$ and an initial trial length $\bar{\alpha} > 0$. Starting from $\alpha = \bar{\alpha}$, we replace α by $\beta\alpha$ until the condition

$$f(\boldsymbol{x}+\alpha\boldsymbol{p}) \le f(\boldsymbol{x}) + c_1\alpha\boldsymbol{g}(\boldsymbol{x})^T\boldsymbol{p}. \tag{4.6}$$

is satisfied. The following result ensures that this procedure terminates and gives a lower bound for the accepted step.

Proposition 4.3.3 (Termination and a step bound)**.** *If $\boldsymbol{g}(\boldsymbol{x})^T\boldsymbol{p} < 0$ and f is differentiable at $\boldsymbol{x}$, backtracking terminates. If the descent lemma applies along all trial segments with a common $L > 0$, then the accepted step satisfies*

$$\alpha \ge \min\left\{\bar{\alpha}, \frac{2\beta(1-c_1)(-\boldsymbol{g}(\boldsymbol{x})^T\boldsymbol{p})}{L\|\boldsymbol{p}\|^2}\right\}. \tag{4.7}$$

Proof. The differentiability expansion proves (4.6) for small enough α, because $\boldsymbol{g}^T\boldsymbol{p} < c_1\boldsymbol{g}^T\boldsymbol{p} < 0$. For the quantitative statement, the descent lemma implies Armijo whenever

$$\alpha \le \tau := \frac{2(1-c_1)(-\boldsymbol{g}^T\boldsymbol{p})}{L\|\boldsymbol{p}\|^2}.$$

If the first trial is accepted, the result follows. Otherwise the previous trial α/β failed and therefore exceeds τ, so $\alpha > \beta\tau$. □

Theorem 4.3.4 (Global stationarity under uniformly positive definite models)**.** *Assume $f \in C^1(\mathbb{R}^n)$ has an L-Lipschitz gradient, $L > 0$, and is bounded below by $f_{\inf} > -\infty$. At each nonstationary iterate solve*

$$B_k\boldsymbol{p}_k = -\boldsymbol{g}_k, \qquad \boldsymbol{g}_k = \boldsymbol{g}(\boldsymbol{x}_k),$$

where B_k is symmetric and $0 < \mu_B I \preceq B_k \preceq M_B I$ for fixed constants. Choose α_k by Armijo backtracking with fixed $c_1, \beta, \bar{\alpha}$. Then

$$\sum_{k=0}^{\infty} \|\boldsymbol{g}_k\|^2 < \infty, \qquad \|\boldsymbol{g}_k\| \longrightarrow 0.$$

Every accumulation point is stationary. If the initial sublevel set is compact, accumulation points exist.

Proof. The step satisfies

$$-\boldsymbol{g}_k^T \boldsymbol{p}_k = \boldsymbol{p}_k^T B_k \boldsymbol{p}_k \geq \mu_B \|\boldsymbol{p}_k\|^2, \qquad -\boldsymbol{g}_k^T \boldsymbol{p}_k = \boldsymbol{g}_k^T B_k^{-1} \boldsymbol{g}_k \geq M_B^{-1} \|\boldsymbol{g}_k\|^2.$$

The backtracking step bound gives $\alpha_k \geq a := \min\{\bar{\alpha}, 2\beta(1-c_1)\mu_B/L\} > 0$. Hence

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1}) \geq \frac{c_1 a}{M_B} \|\boldsymbol{g}_k\|^2.$$

Summing and using the lower bound on f yields the summability claim. Continuity of $\boldsymbol{g}$ makes every accumulation point stationary. Armijo keeps all iterates in the initial sublevel set; compactness then gives a convergent subsequence. □

Remark 4.3.5 (Meaning of "global convergence"). Under its hypotheses, the theorem applies from an arbitrary starting point. We should keep in mind that its conclusion concerns stationarity: it does not establish convergence to a global minimizer, or even convergence of the entire sequence. The global Lipschitz assumption is deliberately simple; it can be weakened to appropriate uniform estimates on a region containing all trial segments. Compactness of the accepted sublevel set alone is not the same as such a trial-segment condition.

4.3.3. Wolfe conditions

For a descent direction, the weak Wolfe conditions combine Armijo decrease with

$$\nabla f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k)^T \boldsymbol{p}_k \geq c_2 \boldsymbol{g}_k^T \boldsymbol{p}_k, \qquad 0 < c_1 < c_2 < 1.$$

The strong Wolfe curvature condition instead requires

$$\left|\nabla f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k)^T \boldsymbol{p}_k\right| \leq c_2 \left|\boldsymbol{g}_k^T \boldsymbol{p}_k\right|.$$

These conditions rule out excessively short steps and are useful for nonlinear CG and BFGS. Simple backtracking need not find a Wolfe step: a practical Wolfe search brackets a suitable interval and interpolates within it. If the line restriction is continuously differentiable and bounded below for positive step lengths, Wolfe steps exist. Objective and gradient evaluations may dominate their cost when each requires a simulation.

4.4. Gradient descent: conditioning and rates

With $B_k = I$ we obtain gradient descent $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \boldsymbol{g}_k$.

Example 4.4.1 (A positive definite quadratic). For $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T A\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}$ with $A \succ 0$, the unique minimizer is $\boldsymbol{x}_* = A^{-1}\boldsymbol{b}$. With a fixed step α and $\boldsymbol{e}_k = \boldsymbol{x}_k - \boldsymbol{x}_*$,

$$\boldsymbol{e}_{k+1} = (I - \alpha A)\boldsymbol{e}_k.$$

An orthonormal eigenbasis shows that convergence from every starting point is equivalent to $|1 - \alpha\lambda_i(A)| < 1$ for every i, that is,

$$0 < \alpha < 2/L, \qquad L = \lambda_{\max}(A).$$

If $\mu = \lambda_{\min}(A)$, the optimal fixed step for the worst-case Euclidean error contraction is

$$\alpha_* = \frac{2}{L+\mu}, \qquad \|I - \alpha_* A\| = \frac{L-\mu}{L+\mu} = \frac{\kappa_2(A)-1}{\kappa_2(A)+1}.$$

Indeed, the maximum of $|1-\alpha\lambda|$ over the eigenvalues is the maximum at the two extreme eigenvalues, and balancing these two endpoint magnitudes gives α_*. Large condition numbers force slow worst-case contraction.

Proposition 4.4.2 (A convex objective-value rate)**.** *Suppose f is convex on $\mathbb{R}^n$, has an L-Lipschitz gradient with $L > 0$, and has a minimizer $\boldsymbol{x}_*$. With $\alpha = 1/L$, gradient descent satisfies, for $k \geq 1$,*

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}_*) \leq \frac{L\,\|\boldsymbol{x}_0 - \boldsymbol{x}_*\|^2}{2k}.$$

Proof. The descent lemma gives $f(\boldsymbol{x}_{j+1}) \leq f(\boldsymbol{x}_j) - \alpha\left\|\boldsymbol{g}_j\right\|^2/2$, so objective values decrease. Expanding squared distances and using convexity,

$$\begin{aligned}\|\boldsymbol{x}_{j+1} - \boldsymbol{x}_*\|^2 &= \|\boldsymbol{x}_j - \boldsymbol{x}_*\|^2 - 2\alpha\boldsymbol{g}_j^T(\boldsymbol{x}_j - \boldsymbol{x}_*) + \alpha^2\left\|\boldsymbol{g}_j\right\|^2 \\ &\leq \|\boldsymbol{x}_j - \boldsymbol{x}_*\|^2 - 2\alpha(f(\boldsymbol{x}_j) - f(\boldsymbol{x}_*)) + \alpha^2\left\|\boldsymbol{g}_j\right\|^2 \\ &\leq \|\boldsymbol{x}_j - \boldsymbol{x}_*\|^2 - 2\alpha(f(\boldsymbol{x}_{j+1}) - f(\boldsymbol{x}_*)).\end{aligned}$$

We sum for $j = 0, \ldots, k-1$ and discard the nonnegative final distance. By monotonicity, the sum of the objective gaps is bounded below by $k(f(\boldsymbol{x}_k) - f(\boldsymbol{x}_*))$, which gives the claim. □

Proposition 4.4.3 (A strongly convex rate)**.** *If, in addition, f is μ-strongly convex, then*

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}_*) \leq (1 - \mu/L)^k\big(f(\boldsymbol{x}_0) - f(\boldsymbol{x}_*)\big).$$

Proof. We minimize the right-hand side of (4.2) over $\boldsymbol{y} \in \mathbb{R}^n$. Since this is a global lower model for f, we obtain

$$f(\boldsymbol{x}_*) \geq f(\boldsymbol{x}) - \frac{1}{2\mu} \|\boldsymbol{g}(\boldsymbol{x})\|^2, \quad \text{so} \quad \|\boldsymbol{g}(\boldsymbol{x})\|^2 \geq 2\mu(f(\boldsymbol{x}) - f(\boldsymbol{x}_*)).$$

Combining this inequality with $f(\boldsymbol{x}_{k+1}) \leq f(\boldsymbol{x}_k) - \|\boldsymbol{g}_k\|^2/(2L)$ and iterating gives the result. □

These rates concern objective values. Error-norm rates and objective-value rates are related under additional curvature bounds, but they should not be identified without explanation.

4.5. Nonlinear conjugate gradients

Nonlinear conjugate gradients combine low storage with information from the preceding search direction. Starting from $\boldsymbol{p}_0 = -\boldsymbol{g}_0$, we set

$$\boldsymbol{p}_{k+1} = -\boldsymbol{g}_{k+1} + \beta_{k+1}\boldsymbol{p}_k.$$

Two common choices are Fletcher–Reeves and Polak–Ribière:

$$\beta_{k+1}^{\mathrm{FR}} = \frac{\boldsymbol{g}_{k+1}^T \boldsymbol{g}_{k+1}}{\boldsymbol{g}_k^T \boldsymbol{g}_k}, \qquad \beta_{k+1}^{\mathrm{PR}} = \frac{\boldsymbol{g}_{k+1}^T (\boldsymbol{g}_{k+1} - \boldsymbol{g}_k)}{\boldsymbol{g}_k^T \boldsymbol{g}_k}.$$

The denominator is used only at nonstationary iterates. On a strictly convex quadratic, with exact line searches, both reduce to linear CG and terminate in at most n steps in exact arithmetic. For nonlinear objectives, conjugacy is lost and these statements no longer apply. A common safeguard replaces β^{PR} by $\max\{0, \beta^{\mathrm{PR}}\}$ and restarts with $-\boldsymbol{g}_{k+1}$ whenever sufficient descent fails. Strong Wolfe searches are commonly used; for Fletcher–Reeves, choosing $c_2 < 1/2$ gives a standard descent safeguard under the usual exact-arithmetic assumptions.

Only a few vectors are stored. This makes the method useful when memory is limited, although a good quasi-Newton method usually behaves better near a solution.

4.6. Newton's method

At $\boldsymbol{x}$, the second-order Taylor model is

$$m_{\boldsymbol{x}}(\boldsymbol{p}) = f(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})^T \boldsymbol{p} + \tfrac{1}{2}\boldsymbol{p}^T \mathcal{H}(\boldsymbol{x})\boldsymbol{p}, \qquad \mathcal{H}(\boldsymbol{x}) = \nabla^2 f(\boldsymbol{x}).$$

Its stationary equation is

$$\mathcal{H}(\boldsymbol{x})\boldsymbol{p} = -\boldsymbol{g}(\boldsymbol{x}). \tag{4.8}$$

If $\mathcal{H}(\boldsymbol{x}) \succ 0$, this gives the unique global minimizer of the model. If $\mathcal{H}(\boldsymbol{x})$ is invertible but indefinite, it gives only a stationary point of the model. In practice, we solve this linear system without computing an inverse explicitly.

Proposition 4.6.1 (Newton descent)**.** *If $\mathcal{H}(\boldsymbol{x}) \succ 0$ and $\boldsymbol{g}(\boldsymbol{x}) \neq 0$, the Newton step is a descent direction, because*

$$\boldsymbol{g}(\boldsymbol{x})^T \boldsymbol{p} = -\boldsymbol{p}^T \mathcal{H}(\boldsymbol{x}) \boldsymbol{p} < 0.$$

Example 4.6.2 (A Newton ascent direction). Let us take $f(x) = x^4 - x^2$ and $x = 0.1$. We then have $g = -0.196$, $\mathcal{H} = -1.88$ and $p = -g/\mathcal{H} \approx -0.104255$, so $gp > 0$. The usual small-step Armijo guarantee does not apply. Shrinking the step alone does not turn an ascent direction into a descent direction.

4.6.1. Local quadratic convergence

Definition 4.6.3. For an iteration converging to $\boldsymbol{x}_*$, write $\boldsymbol{e}_k = \boldsymbol{x}_k - \boldsymbol{x}_*$. It converges Q-linearly if $\|\boldsymbol{e}_{k+1}\| \leq q \|\boldsymbol{e}_k\|$ eventually for some $q < 1$, Q-superlinearly if $\|\boldsymbol{e}_{k+1}\| / \|\boldsymbol{e}_k\| \to 0$ (unless the solution is reached finitely), and Q-quadratically if $\|\boldsymbol{e}_{k+1}\| \leq C \|\boldsymbol{e}_k\|^2$ eventually for some finite C.

Theorem 4.6.4 (Newton local convergence)**.** *Let $f \in C^2$ near a stationary point $\boldsymbol{x}_*$, and suppose $\mathcal{H}_* := \mathcal{H}(\boldsymbol{x}_*)$ is nonsingular. Assume $\mathcal{H}$ is Lipschitz on a neighbourhood of $\boldsymbol{x}_*$, with constant $L_{\mathcal{H}}$. There exist $\delta, C > 0$ such that, for $\|\boldsymbol{x}_0 - \boldsymbol{x}_*\| < \delta$, the full-step Newton iteration is well defined, stays in that neighbourhood, and satisfies*

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_*\| \leq C \|\boldsymbol{x}_k - \boldsymbol{x}_*\|^2 .$$

If $\mathcal{H}_* \succ 0$, the limit is a strict local minimizer.

Proof. Continuity and nonsingularity imply that, on a sufficiently small closed ball about $\boldsymbol{x}_*$, all $\mathcal{H}(\boldsymbol{x})$ are invertible with $\left\|\mathcal{H}(\boldsymbol{x})^{-1}\right\| \leq M$. This follows, for instance, by applying the Neumann series to $\mathcal{H}(\boldsymbol{x}) = \mathcal{H}_*[I + \mathcal{H}_*^{-1}(\mathcal{H}(\boldsymbol{x}) - \mathcal{H}_*)]$.

We set $\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x}_*$ and $\boldsymbol{x}_+ = \boldsymbol{x} - \mathcal{H}(\boldsymbol{x})^{-1} \boldsymbol{g}(\boldsymbol{x})$. Since $\boldsymbol{g}(\boldsymbol{x}_*) = 0$,

$$\boldsymbol{g}(\boldsymbol{x}) = \int_0^1 \mathcal{H}(\boldsymbol{x}_* + t\boldsymbol{e}) \boldsymbol{e} \, dt.$$

Consequently,

$$\boldsymbol{x}_+ - \boldsymbol{x}_* = \mathcal{H}(\boldsymbol{x})^{-1} \int_0^1 [\mathcal{H}(\boldsymbol{x}) - \mathcal{H}(\boldsymbol{x}_* + t\boldsymbol{e})] \boldsymbol{e} \, dt,$$

$$\|\boldsymbol{x}_+ - \boldsymbol{x}_*\| \leq M L_{\mathcal{H}} \int_0^1 (1 - t) \|\boldsymbol{e}\|^2 \, dt = \frac{M L_{\mathcal{H}}}{2} \|\boldsymbol{e}\|^2 .$$

We choose $C \geq M L_{\mathcal{H}}/2$ positive and δ small enough that $C\delta < 1$ and the closed δ-ball lies in the chosen neighbourhood. The estimate maps this ball into itself and contracts errors. Induction establishes existence and convergence, and the same estimate gives quadratic convergence. Positive definiteness at the limit invokes the second-order sufficiency result. □

Remark 4.6.5. The theorem is fundamentally a root-finding result for $\nabla f = 0$. Nonsingularity alone allows convergence to a maximum or saddle. It is also local: it makes no guarantee from a distant starting point.

Proposition 4.6.6 (Eventual acceptance of full Newton steps)**.** *Under the hypotheses of the local Newton theorem with $\mathcal{H}_* \succ 0$, Armijo backtracking starting at $\bar{\alpha} = 1$ accepts $\alpha = 1$ whenever $\boldsymbol{x}$ is sufficiently close to $\boldsymbol{x}_*$ and $c_1 < 1/2$.*

Proof. Near $\boldsymbol{x}_*$, $\mathcal{H}(\boldsymbol{x}) \succeq \mu I$ for some $\mu > 0$. The Lipschitz Hessian gives a Taylor remainder bounded by $L_{\mathcal{H}} \|\boldsymbol{p}\|^3 /6$. For the Newton step, $\boldsymbol{g}^T\boldsymbol{p} = -\boldsymbol{p}^T\mathcal{H}\boldsymbol{p}$, whence

$$f(\boldsymbol{x}+\boldsymbol{p}) - f(\boldsymbol{x}) - c_1\boldsymbol{g}^T\boldsymbol{p} \le -(1/2 - c_1)\mu \|\boldsymbol{p}\|^2 + \frac{L_{\mathcal{H}}}{6}\|\boldsymbol{p}\|^3.$$

The right side is nonpositive for sufficiently small $\boldsymbol{p}$. The Newton step tends to zero as $\boldsymbol{x} \to \boldsymbol{x}_*$, proving the claim. □

Far from the solution, a modified Hessian $B_k \succ 0$ or a trust region can supply useful steps. The global stationarity result applies to modified Newton matrices only if its uniform upper and lower spectral bounds and smoothness hypotheses hold.

Example 4.6.7 (A comparison on a quadratic). For $f(x, y) = \frac{1}{2}(x^2 + 10y^2)$,

$$\boldsymbol{g} = (x, 10y)^T, \qquad \mathcal{H} = \operatorname{diag}(1, 10), \qquad \boldsymbol{p}_{\mathrm{N}} = (-x, -y)^T.$$

A full Newton step reaches $(0, 0)$ exactly. Gradient descent with $\alpha = 0.1$ sends (x, y) to $(0.9x, 0)$ and thereafter contracts the first coordinate by 0.9 at every step.

4.6.2. Modified and truncated Newton methods

Far from a minimizer, we can use $B_k = \nabla^2 f(\boldsymbol{x}_k) + \tau_k I \succ 0$, a modified Cholesky factorization, or a trust-region subproblem. A positive shift must exceed the magnitude of any negative eigenvalue; an arbitrary small shift is insufficient. Global stationarity under the earlier theorem additionally requires uniform spectral bounds.

Dense Hessian storage costs $O(n^2)$ and factorization $O(n^3)$. Truncated Newton methods use Krylov iterations and Hessian–vector products. An inexact solve can enforce

$$\|\nabla^2 f(\boldsymbol{x}_k)\boldsymbol{p}_k + \boldsymbol{g}_k\| \le \eta_k \|\boldsymbol{g}_k\|.$$

Near a minimizer with nonsingular Lipschitz Hessian, full steps and sufficiently small bounded forcing terms yield local linear convergence; $\eta_k \to 0$ yields superlinear convergence, and $\eta_k = O(\|\boldsymbol{g}_k\|)$ yields a quadratic bound. Far away, negative curvature and descent still need safeguards. Inner solves should become more accurate as the outer iteration approaches the solution.

4.7. Quasi-Newton methods

We can also approximate curvature using successive iterates and gradients, without evaluating the Hessian directly. This leads to quasi-Newton methods. We define

$$\boldsymbol{s}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k, \qquad \boldsymbol{y}_k = \boldsymbol{g}_{k+1} - \boldsymbol{g}_k.$$

The secant equation is $B_{k+1}\boldsymbol{s}_k = \boldsymbol{y}_k$. Indeed,

$$\boldsymbol{y}_k = \int_0^1 \nabla^2 f(\boldsymbol{x}_k + t\boldsymbol{s}_k)\boldsymbol{s}_k \, dt,$$

so it records an averaged Hessian action rather than a complete Hessian.

4.7.1. BFGS

We write H_k for an inverse-Hessian approximation; this differs from the exact Hessian $\mathcal{H}(\boldsymbol{x}) = \nabla^2 f(\boldsymbol{x})$ used above. BFGS computes $\boldsymbol{p}_k = -H_k\boldsymbol{g}_k$ and updates

$$H_{k+1} = (I - \rho_k \boldsymbol{s}_k \boldsymbol{y}_k^T) H_k (I - \rho_k \boldsymbol{y}_k \boldsymbol{s}_k^T) + \rho_k \boldsymbol{s}_k \boldsymbol{s}_k^T, \qquad \rho_k = \frac{1}{\boldsymbol{y}_k^T \boldsymbol{s}_k}.$$

It satisfies $H_{k+1}\boldsymbol{y}_k = \boldsymbol{s}_k$. If $H_k \succ 0$ and $\boldsymbol{y}_k^T \boldsymbol{s}_k > 0$, then $H_{k+1} \succ 0$: for a nonzero vector $\boldsymbol{v}$, its quadratic form is the sum of

$$\big((I - \rho_k \boldsymbol{y}_k \boldsymbol{s}_k^T)\boldsymbol{v}\big)^T H_k \big((I - \rho_k \boldsymbol{y}_k \boldsymbol{s}_k^T)\boldsymbol{v}\big) \quad \text{and} \quad \rho_k (\boldsymbol{s}_k^T \boldsymbol{v})^2,$$

which cannot both vanish. A weak Wolfe step gives

$$\boldsymbol{y}_k^T \boldsymbol{s}_k \geq \alpha_k (c_2 - 1) \boldsymbol{g}_k^T \boldsymbol{p}_k > 0.$$

Very small curvature products still require numerical safeguards, such as damping or skipping an update. BFGS uses $O(n^2)$ storage and work per iteration. Under standard local regularity assumptions, including a positive-definite Lipschitz Hessian near the minimizer, sufficiently close initialization and a suitable Wolfe search accepting eventual full steps, it has superlinear local convergence. This is a local statement, separate from the uniform spectral bounds used in the earlier stationarity theorem.

4.7.2. Limited-memory BFGS

L-BFGS retains only the most recent ℓ pairs $(\boldsymbol{s}_j, \boldsymbol{y}_j)$. We use ℓ for the memory length so that m continues to denote the number of residuals. A two-loop recursion applies the corresponding inverse-Hessian approximation without storing a dense matrix, requiring $O(\ell n)$ work and storage. A common initial scaling is

$$H_k^{(0)} = \frac{\boldsymbol{s}_{k-1}^T \boldsymbol{y}_{k-1}}{\boldsymbol{y}_{k-1}^T \boldsymbol{y}_{k-1}} I,$$

when the curvature product is positive. L-BFGS is useful for very large problems. Fixed limited memory does not automatically inherit full BFGS's superlinear convergence.

4.8. The structure of nonlinear least squares

We now consider objectives with a sum-of-squares structure. Let $\boldsymbol{r} : \mathbb{R}^n \to \mathbb{R}^m$ be a residual map, and define

$$f(\boldsymbol{x}) = \tfrac{1}{2} \|\boldsymbol{r}(\boldsymbol{x})\|^2 = \tfrac{1}{2} \sum_{i=1}^{m} r_i(\boldsymbol{x})^2. \tag{4.9}$$

For parameter estimation, $r_i(\boldsymbol{x}) = m(t_i; \boldsymbol{x}) - \boldsymbol{y}_i$ compares a model with an observation. Changing the sign of every residual leaves f and the resulting Gauss–Newton equations unchanged, provided the Jacobian sign changes consistently.

4.8.1. The linear case and residual orthogonality

If $\boldsymbol{r}(\boldsymbol{x}) = A\boldsymbol{x} - \boldsymbol{b}$, stationarity is $A^T(A\boldsymbol{x} - \boldsymbol{b}) = 0$. Every minimizer satisfies these normal equations, and conversely every solution minimizes the convex quadratic. To see the converse directly, let $\boldsymbol{r}_* = A\boldsymbol{x}_* - \boldsymbol{b}$ with $A^T\boldsymbol{r}_* = 0$. Then

$$\|A(\boldsymbol{x}_* + \boldsymbol{p}) - \boldsymbol{b}\|^2 = \|\boldsymbol{r}_*\|^2 + \|A\boldsymbol{p}\|^2 .$$

Full column rank makes the minimizer unique; otherwise the minimizers differ by elements of $\ker A$. The residual $\boldsymbol{r}_*$ is perpendicular to $\operatorname{range}(A)$.

4.8.2. Gradient and Hessian: a derivation

Write

$$J(\boldsymbol{x}) = D\boldsymbol{r}(\boldsymbol{x}) \in \mathbb{R}^{m \times n}, \qquad J_{ij}(\boldsymbol{x}) = \frac{\partial r_i}{\partial \boldsymbol{x}_j}(\boldsymbol{x}).$$

Proposition 4.8.1 (Least-squares derivatives)**.** *If $\boldsymbol{r} \in C^1$, then $\nabla f(\boldsymbol{x}) = J(\boldsymbol{x})^T \boldsymbol{r}(\boldsymbol{x})$. If $\boldsymbol{r} \in C^2$, then*

$$\nabla^2 f(\boldsymbol{x}) = J(\boldsymbol{x})^T J(\boldsymbol{x}) + S(\boldsymbol{x}), \qquad S(\boldsymbol{x}) = \sum_{i=1}^{m} r_i(\boldsymbol{x}) \nabla^2 r_i(\boldsymbol{x}). \tag{4.10}$$

Proof. For an increment $\boldsymbol{p}$, the chain rule gives

$$Df(\boldsymbol{x})[\boldsymbol{p}] = \langle \boldsymbol{r}(\boldsymbol{x}), J(\boldsymbol{x})\boldsymbol{p} \rangle = \left\langle J(\boldsymbol{x})^T \boldsymbol{r}(\boldsymbol{x}), \boldsymbol{p} \right\rangle .$$

For increments $\boldsymbol{u}, \boldsymbol{v}$, differentiating again gives

$$D^2 f(\boldsymbol{x})[\boldsymbol{u}, \boldsymbol{v}] = \langle J(\boldsymbol{x})\boldsymbol{u}, J(\boldsymbol{x})\boldsymbol{v} \rangle + \sum_{i=1}^{m} r_i(\boldsymbol{x}) D^2 r_i(\boldsymbol{x})[\boldsymbol{u}, \boldsymbol{v}],$$

which identifies the Hessian as (4.10). □

At a stationary point,

$$J(\boldsymbol{x}_*)^T \boldsymbol{r}(\boldsymbol{x}_*) = 0 \quad \Longleftrightarrow \quad \boldsymbol{r}(\boldsymbol{x}_*) \perp \operatorname{range}(J(\boldsymbol{x}_*)).$$

This expresses orthogonality to the *linearized* residual directions. If J has full column rank locally, the image can locally be parametrized as an immersed n-dimensional manifold, and the columns of J span its tangent space. The algebraic orthogonality statement does not require a manifold interpretation.

Corollary 4.8.2 (An exact fit with identifiable linearization)**.** *If $\boldsymbol{r} \in C^2$, $\boldsymbol{r}(\boldsymbol{x}_*) = 0$ and $J(\boldsymbol{x}_*)$ has full column rank, then $\boldsymbol{x}_*$ is a strict local minimizer. It is also a global minimizer, since $f \geq 0 = f(\boldsymbol{x}_*)$.*

Proof. The gradient vanishes and the Hessian equals $J_*^T J_* \succ 0$. Apply the second-order sufficiency result. There may still be other global minimizers away from $\boldsymbol{x}_*$. □

4.8.3. Weighted least squares

When observations have different uncertainties or correlated errors, use

$$f(\boldsymbol{x}) = \tfrac{1}{2}\boldsymbol{r}(\boldsymbol{x})^T W \boldsymbol{r}(\boldsymbol{x}), \qquad W \succ 0.$$

For a fixed symmetric weight matrix,

$$\nabla f = J^T W \boldsymbol{r}, \qquad \nabla^2 f = J^T W J + \sum_{i=1}^{m} (W\boldsymbol{r})_i \nabla^2 r_i.$$

If $W = C^T C$, the transformed residual $C\boldsymbol{r}$ and Jacobian CJ reduce the problem to ordinary least squares. These formulas require modification when W depends on the parameters. Changing residual weights changes the fitting objective, whereas scaling parameter coordinates changes its representation.

Example 4.8.3 (Exponential fitting). Let $\boldsymbol{x} = (a, b)^T$ and $r_i(a, b) = ae^{bt_i} - \boldsymbol{y}_i$. Then

$$J_i(a, b) = \begin{pmatrix} e^{bt_i} & at_i e^{bt_i} \end{pmatrix}, \qquad \nabla^2 r_i(a, b) = \begin{pmatrix} 0 & t_i e^{bt_i} \\ t_i e^{bt_i} & at_i^2 e^{bt_i} \end{pmatrix}. \tag{4.11}$$

For $a \neq 0$ and at least two distinct times, J has rank two: a linear combination of its columns vanishes exactly when $\boldsymbol{u} + \boldsymbol{v} a t_i = 0$ for every i, forcing $\boldsymbol{u} = \boldsymbol{v} = 0$. At $a = 0$ the second column vanishes, so the rate parameter is invisible to the first-order model. Closely clustered times can make the columns nearly dependent even when they are independent in exact arithmetic.

4.9. Gauss–Newton: derivation, geometry and convergence

4.9.1. Two derivations of the same step

Applying Newton's method to (4.9) gives the linear system

$$(J^T J + S)\boldsymbol{p} = -J^T \boldsymbol{r}.$$

Dropping S gives the *Gauss–Newton equations*

$$J^T J\boldsymbol{p} = -J^T \boldsymbol{r}. \tag{4.12}$$

We can obtain the same step by linearizing the residual itself:

$$\boldsymbol{r}(\boldsymbol{x}+\boldsymbol{p}) = \boldsymbol{r}(\boldsymbol{x}) + J(\boldsymbol{x})\boldsymbol{p} + O(\|\boldsymbol{p}\|^2)$$

when $\boldsymbol{r}$ has locally Lipschitz first derivative. We then minimize the resulting model

$$q_{\boldsymbol{x}}(\boldsymbol{p}) = \tfrac{1}{2}\left\|\boldsymbol{r}(\boldsymbol{x}) + J(\boldsymbol{x})\boldsymbol{p}\right\|^2. \tag{4.13}$$

Its normal equations are exactly (4.12). Thus Gauss–Newton repeatedly solves a *linear least-squares problem in the increment.*

Proposition 4.9.1 (Existence, uniqueness and descent)**.** *For any matrix J and vector $\boldsymbol{r}$, the subproblem* (4.13) *has a minimizer. The minimizer is unique if and only if J has full column rank. Every minimizing step $\boldsymbol{p}$ satisfies*

$$\boldsymbol{g}^T\boldsymbol{p} = -\left\|J\boldsymbol{p}\right\|^2, \qquad \boldsymbol{g} = J^T\boldsymbol{r}. \tag{4.14}$$

If $\boldsymbol{g} \neq 0$, every such step is a descent direction, even if J is rank deficient.

Proof. The vector $J\boldsymbol{p}$ ranges over the closed linear subspace range(J). We obtain an attainable minimizer by projecting $-\boldsymbol{r}$ orthogonally onto that subspace. All solutions differ by a vector in $\ker J$, proving the uniqueness claim. Multiplying $J^T(J\boldsymbol{p}+\boldsymbol{r}) = 0$ by $\boldsymbol{p}^T$ gives (4.14). If $J\boldsymbol{p} = 0$, the normal equations force $\boldsymbol{g} = 0$, so $\boldsymbol{g} \neq 0$ makes the right side strictly negative. □

The predicted residual $\boldsymbol{r} + J\boldsymbol{p}$ is perpendicular to range(J). It is the projection of $\boldsymbol{r}$ onto range$(J)^\perp$. The actual nonlinear residual $\boldsymbol{r}(\boldsymbol{x}+\boldsymbol{p})$ generally differs from that projection.

A rank-deficient problem admits arbitrarily large null-space additions to its minimizing steps. Descent alone therefore does not establish a useful global convergence theorem. A minimum-norm solution, explicit rank management or regularization is needed in a reliable implementation. With full rank, the step is

$$\boldsymbol{p}(\boldsymbol{x}) = -A(\boldsymbol{x})^{-1}\boldsymbol{g}(\boldsymbol{x}), \qquad A(\boldsymbol{x}) = J(\boldsymbol{x})^T J(\boldsymbol{x}).$$

If $A(\boldsymbol{x}_k)$ has uniform positive lower and finite upper spectral bounds, the line-search iteration fits the global stationarity result, provided its other assumptions hold.

4.9.2. Characterization of the local behavior for Gauss-Newton

To study the local behaviour, we need to compare the size of S with that of A. A small residual is useful only when the second derivatives of the residuals and the conditioning are also controlled.

Theorem 4.9.2 (Local Gauss–Newton convergence)**.** *Suppose $\boldsymbol{r} \in C^2$ on a neighbourhood of $\boldsymbol{x}_*$, $J_* = J(\boldsymbol{x}_*)$ has full column rank and $\boldsymbol{g}(\boldsymbol{x}_*) = J_*^T \boldsymbol{r}(\boldsymbol{x}_*) = 0$. Put $A_* = J_*^T J_*$ and $S_* = S(\boldsymbol{x}_*)$. Define*

$$\|\boldsymbol{v}\|_{A_*} = \sqrt{\boldsymbol{v}^T A_* \boldsymbol{v}}, \qquad \gamma = \left\| A_*^{-1/2} S_* A_*^{-1/2} \right\|.$$

If $\gamma < 1$, then for every $q \in (\gamma, 1)$ and all sufficiently close starting points, full-step Gauss–Newton is well defined and converges to $\boldsymbol{x}_*$ with

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_*\|_{A_*} \le q \, \|\boldsymbol{x}_k - \boldsymbol{x}_*\|_{A_*} . \tag{4.15}$$

Moreover, $\boldsymbol{x}_*$ is a strict local minimizer.

Proof. Full rank persists locally by continuity, so

$$T(\boldsymbol{x}) = \boldsymbol{x} - A(\boldsymbol{x})^{-1} \boldsymbol{g}(\boldsymbol{x})$$

is C^1 near $\boldsymbol{x}_*$. Since $\boldsymbol{g}(\boldsymbol{x}_*) = 0$, differentiation cancels the term involving the derivative of A^{-1} and yields

$$DT(\boldsymbol{x}_*) = I - A_*^{-1} \nabla^2 f(\boldsymbol{x}_*) = -A_*^{-1} S_*. \tag{4.16}$$

For any linear map M, its induced A_*-norm is $\|M\|_{A_*} = \left\| A_*^{1/2} M A_*^{-1/2} \right\|$.

Thus $\|DT(\boldsymbol{x}_*)\|_{A_*} = \gamma$. Continuity gives a convex A_*-norm ball about $\boldsymbol{x}_*$ where $\|DT(\boldsymbol{x})\|_{A_*} \le q$. Integrating the derivative along the segment from $\boldsymbol{x}_*$ to $\boldsymbol{x}$ gives

$$\|T(\boldsymbol{x}) - \boldsymbol{x}_*\|_{A_*} \le q \, \|\boldsymbol{x} - \boldsymbol{x}_*\|_{A_*} .$$

The ball is invariant, and induction proves convergence and (4.15). Finally,

$$\nabla^2 f(\boldsymbol{x}_*) = A_*^{1/2} \big(I + A_*^{-1/2} S_* A_*^{-1/2} \big) A_*^{1/2} \succ 0,$$

because the symmetric middle perturbation has norm below one. The second-order sufficiency result then proves that the limit is a strict local minimizer. □

Remark 4.9.3. The contraction is in the fixed norm $\|\cdot\|_{A_*}$. Norm equivalence gives geometric error bounds in the Euclidean norm, but does not automatically give the same one-step Euclidean contraction factor. Since the normalized S_* is symmetric, γ also equals the spectral radius of $DT(\boldsymbol{x}_*)$. The condition is sufficient for local attraction; no claim is made at the boundary $\gamma = 1$.

Corollary 4.9.4. *Under the differentiability and rank hypotheses of the local Gauss–Newton theorem, its contraction assumption holds if*

$$\sum_{i=1}^{m} |r_i(\boldsymbol{x}_*)| \, \left\| \nabla^2 r_i(\boldsymbol{x}_*) \right\| < \sigma_{\min}(J_*)^2. \tag{4.17}$$

Proof. The triangle inequality gives $\|S_*\| \le \sum_i |r_i(\boldsymbol{x}_*)| \left\|\nabla^2 r_i(\boldsymbol{x}_*)\right\|$. Also,

$$\gamma \le \left\|A_*^{-1/2}\right\|^2 \|S_*\| = \frac{\|S_*\|}{\sigma_{\min}(J_*)^2}.$$

□

A large residual does not force failure: affine residuals have $S = 0$ regardless of residual size. Conversely, a small numerical residual can be insufficient when J is nearly rank deficient or second derivatives are large.

Theorem 4.9.5 (Quadratic convergence at a zero residual)**.** *Suppose $\boldsymbol{r} \in C^1$ near $\boldsymbol{x}_*$, $\boldsymbol{r}(\boldsymbol{x}_*) = 0$, J_* has full column rank and J is Lipschitz near $\boldsymbol{x}_*$. Then full-step Gauss–Newton converges Q-quadratically in the Euclidean norm from sufficiently close starting points.*

Proof. On a small ball, $J(\boldsymbol{x})$ retains full rank. Its left pseudoinverse is

$$J(\boldsymbol{x})^\dagger = (J(\boldsymbol{x})^T J(\boldsymbol{x}))^{-1} J(\boldsymbol{x})^T,$$

and satisfies $\left\|J(\boldsymbol{x})^\dagger\right\| \le M$ for a uniform constant M. For $\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x}_*$, the integral formula gives

$$\boldsymbol{r}(\boldsymbol{x}) = \int_0^1 J(\boldsymbol{x}_* + t\boldsymbol{e})\boldsymbol{e}\, dt = J(\boldsymbol{x})\boldsymbol{e} + \boldsymbol{d}(\boldsymbol{x}), \quad \|\boldsymbol{d}(\boldsymbol{x})\| \le \frac{L_J}{2}\|\boldsymbol{e}\|^2.$$

Since $J(\boldsymbol{x})^\dagger J(\boldsymbol{x}) = I$,

$$T(\boldsymbol{x}) - \boldsymbol{x}_* = -J(\boldsymbol{x})^\dagger \boldsymbol{d}(\boldsymbol{x}), \qquad \|T(\boldsymbol{x}) - \boldsymbol{x}_*\| \le \frac{ML_J}{2}\|\boldsymbol{e}\|^2.$$

A sufficiently small invariant ball gives convergence by the same induction as in the local Newton theorem. □

Merely observing $DT(\boldsymbol{x}_*) = 0$ for a differentiable iteration yields a superlinear estimate; the explicit Lipschitz argument above supplies the stronger quadratic bound.

4.10. Numerical linear algebra and scaling

Normal equations are useful for derivation, but explicitly forming $J^T J$ may degrade numerical accuracy.

Proposition 4.10.1. *For full-column-rank J, define $\kappa_2(J) = \sigma_{\max}(J)/\sigma_{\min}(J)$. Then*

$$\kappa_2(J^T J) = \kappa_2(J)^2.$$

Proof. The eigenvalues of $J^T J$ are the squared singular values of J. Their largest-to-smallest ratio gives the identity. □

For a thin QR factorization $J = QR$, with $Q^TQ = I$ and invertible upper triangular R,

$$\|J\boldsymbol{p} + \boldsymbol{r}\|^2 = \left\|R\boldsymbol{p} + Q^T\boldsymbol{r}\right\|^2 + \left\|(I - QQ^T)\boldsymbol{r}\right\|^2.$$

We can therefore compute the step by solving $R\boldsymbol{p} = -Q^T\boldsymbol{r}$, without forming a Gram matrix or an inverse. If rank is deficient or uncertain, an SVD $J = U\Sigma V^T$ reveals small singular values. The minimum-norm minimizing step is $\boldsymbol{p} = -J^\dagger\boldsymbol{r}$. Numerical truncation of small singular values is a modelling/regularization decision, not an exact identity for the original full-rank problem.

For a nonsingular parameter transformation $\boldsymbol{x} = D\boldsymbol{z}$, the residual Jacobian in $\boldsymbol{z}$ is $J(\boldsymbol{x})D$. Full-rank, undamped Gauss–Newton respects this change of coordinates in exact arithmetic: its unique minimizing increments correspond through $\boldsymbol{p}_{\boldsymbol{x}} = D\boldsymbol{p}_{\boldsymbol{z}}$. Conditioning and damping, however, depend on the representation. A penalty $\lambda\|\boldsymbol{p}_{\boldsymbol{z}}\|^2$ becomes $\lambda\left\|D^{-1}\boldsymbol{p}_{\boldsymbol{x}}\right\|^2$.

4.11. Trust-region methods

A general trust-region method minimizes a quadratic model within a neighborhood:

$$\min_{\|\boldsymbol{p}\| \le \Delta_k} m_k(\boldsymbol{p}), \qquad m_k(\boldsymbol{p}) = f(\boldsymbol{x}_k) + \boldsymbol{g}_k^T\boldsymbol{p} + \tfrac{1}{2}\boldsymbol{p}^T B_k\boldsymbol{p}.$$

The symmetric matrix B_k may be indefinite. For a trial step with positive predicted decrease, we form the ratio

$$\rho_k = \frac{f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k + \boldsymbol{p}_k)}{m_k(\boldsymbol{0}) - m_k(\boldsymbol{p}_k)}.$$

We accept a step when the agreement is adequate. Otherwise, we reject it and reduce Δ_k. Strong agreement, especially at the boundary, can justify enlarging the radius. A zero predicted decrease requires a stationarity check or another step, rather than division by zero.

The subproblem need not be solved exactly. For $\boldsymbol{g}_k \neq 0$, the Cauchy step is $-t_k\boldsymbol{g}_k$, where

$$t_k = \begin{cases} \min\{\|\boldsymbol{g}_k\|^2/(\boldsymbol{g}_k^T B_k\boldsymbol{g}_k),\ \Delta_k/\|\boldsymbol{g}_k\|\}, & \boldsymbol{g}_k^T B_k\boldsymbol{g}_k > 0, \\ \Delta_k/\|\boldsymbol{g}_k\|, & \boldsymbol{g}_k^T B_k\boldsymbol{g}_k \le 0. \end{cases}$$

It gives a basic amount of model decrease. Dogleg methods join a gradient step to the Newton step when $B_k \succ 0$. Truncated CG follows approximate Newton corrections until convergence, a boundary intersection, or negative curvature is detected. This allows large-scale methods based on Hessian–vector products. At a stationary saddle, a negative-curvature direction is needed to improve beyond the zero Cauchy step.

Trust regions handle indefinite curvature naturally. Global stationarity results require model bounds, adequate subproblem decrease, and specified acceptance and radius rules; the ratio test alone is not a convergence theorem. The least-squares specialization and its link to Levenberg–Marquardt are developed below.

4.12. Levenberg–Marquardt

4.12.1. A regularized linearized problem

For $\lambda > 0$, the basic Levenberg–Marquardt (LM) step minimizes

$$q_{\boldsymbol{x}}(\boldsymbol{p}) + \frac{\lambda}{2}\|\boldsymbol{p}\|^2 = \frac{1}{2}\|\boldsymbol{r} + J\boldsymbol{p}\|^2 + \frac{\lambda}{2}\|\boldsymbol{p}\|^2. \tag{4.18}$$

Its normal equations are

$$(A + \lambda I)\boldsymbol{p} = -\boldsymbol{g}, \qquad A = J^T J, \quad \boldsymbol{g} = J^T \boldsymbol{r}. \tag{4.19}$$

Proposition 4.12.1 (Well-posedness and descent)**.** *For every $J, \boldsymbol{r}$ and $\lambda > 0$,* (4.18) *has a unique minimizer. If $\boldsymbol{g} \neq 0$, it is a descent direction:*

$$\boldsymbol{g}^T \boldsymbol{p} = -\boldsymbol{p}^T (A + \lambda I)\boldsymbol{p} < 0.$$

Moreover, $\|\boldsymbol{p}\| \leq \|\boldsymbol{g}\| / \lambda$.

Proof. The Hessian $A + \lambda I$ is positive definite with smallest eigenvalue at least λ. Strict convexity, the linear system and its inverse bound prove the assertions. □

For fixed $\boldsymbol{x}$, as $\lambda \to \infty$, a Neumann-series expansion gives

$$\boldsymbol{p} = -\lambda^{-1}(I + \lambda^{-1}A)^{-1}\boldsymbol{g} = -\lambda^{-1}\boldsymbol{g} + O(\lambda^{-2}).$$

If A is invertible, $\boldsymbol{p} \to -A^{-1}\boldsymbol{g}$ as $\lambda \downarrow 0$. If J is rank deficient, $\boldsymbol{g} \in \operatorname{range}(A)$ and a spectral decomposition shows that the limit is the minimum-norm step $-J^{\dagger}\boldsymbol{r}$; components in $\ker A$ are zero.

The penalty is on the current *increment*. It does not automatically mean that the original objective has acquired a parameter penalty $\lambda\|\boldsymbol{x}\|^2/2$. More generally, keeping the notation of the quadratic model, we can use

$$(J_k^T J_k + \lambda_k D_k)\boldsymbol{p}_k = -\boldsymbol{g}_k, \qquad D_k \succ 0.$$

The penalty is $\lambda_k \boldsymbol{p}^T D_k \boldsymbol{p}/2$. If $C_k^T C_k = D_k$, the scaled trust region is $\|C_k \boldsymbol{p}\| \leq \Delta_k$. A diagonal choice needs a positive floor when some Jacobian columns vanish.

4.12.2. The trust-region subproblem and its multiplier

Let $\Delta > 0$. We consider the constrained model problem

$$\min_{\|\boldsymbol{p}\| \leq \Delta} q_{\boldsymbol{x}}(\boldsymbol{p}). \tag{4.20}$$

Theorem 4.12.2 (Characterization of the least-squares trust-region step)**.** *A vector $\boldsymbol{p}_*$ solves* (4.20) *if and only if there exists $\lambda \geq 0$ such that*

$$(A + \lambda I)\boldsymbol{p}_* = -\boldsymbol{g}, \qquad \|\boldsymbol{p}_*\| \leq \Delta, \qquad \lambda(\|\boldsymbol{p}_*\|^2 - \Delta^2) = 0. \tag{4.21}$$

A solution exists. For a positive multiplier the associated step is unique.

Proof. Existence follows from compactness of the ball. For necessity, an interior minimizer has $\nabla q_{\boldsymbol{x}}(\boldsymbol{p}_*) = 0$, and we choose $\lambda = 0$. At a boundary minimizer, convexity of the feasible set and directional necessity give

$$\nabla q_{\boldsymbol{x}}(\boldsymbol{p}_*)^T(\boldsymbol{v} - \boldsymbol{p}_*) \geq 0 \quad \text{for every } \|\boldsymbol{v}\| \leq \Delta.$$

Thus $\boldsymbol{p}_*$ minimizes the linear functional $\nabla q_{\boldsymbol{x}}(\boldsymbol{p}_*)^T\boldsymbol{v}$ on the ball. If this gradient is nonzero, its unique minimizer is $-\Delta\nabla q_{\boldsymbol{x}}(\boldsymbol{p}_*)/\|\nabla q_{\boldsymbol{x}}(\boldsymbol{p}_*)\|$, so $\nabla q_{\boldsymbol{x}}(\boldsymbol{p}_*) = -\lambda\boldsymbol{p}_*$ for $\lambda = \|\nabla q_{\boldsymbol{x}}(\boldsymbol{p}_*)\|/\Delta > 0$. If it is zero, choose $\lambda = 0$. This proves (4.21).

Conversely, suppose (4.21) holds. For any feasible $\boldsymbol{p}$ and $\boldsymbol{d} = \boldsymbol{p} - \boldsymbol{p}_*$,

$$q_{\boldsymbol{x}}(\boldsymbol{p}) - q_{\boldsymbol{x}}(\boldsymbol{p}_*) = \tfrac{1}{2}\boldsymbol{d}^T(A + \lambda I)\boldsymbol{d} + \tfrac{\lambda}{2}(\|\boldsymbol{p}_*\|^2 - \|\boldsymbol{p}\|^2) \geq 0.$$

The first term is nonnegative. The second is zero if $\lambda = 0$, and otherwise nonnegative by complementarity and feasibility. Positive λ makes the first quadratic form strictly positive for $\boldsymbol{d} \neq 0$, proving uniqueness. □

This gives a precise relationship between trust regions and LM: the multiplier depends on the radius and the current model. It is not generally a fixed constant, and the multiplier may be zero when an unconstrained model minimizer is feasible.

4.12.3. Model agreement and acceptance

To assess the agreement between the model and the objective, we define the predicted and actual decreases as

$$\text{pred} = q_{\boldsymbol{x}}(0) - q_{\boldsymbol{x}}(\boldsymbol{p}), \qquad \text{ared} = f(\boldsymbol{x}) - f(\boldsymbol{x} + \boldsymbol{p}).$$

For an LM step with $\boldsymbol{g} \neq 0$,

$$\text{pred} = -\boldsymbol{g}^T\boldsymbol{p} - \tfrac{1}{2}\boldsymbol{p}^T A\boldsymbol{p} = \tfrac{1}{2}\boldsymbol{p}^T A\boldsymbol{p} + \lambda\|\boldsymbol{p}\|^2 > 0.$$

A trust-region implementation forms $\rho = \text{ared} / \text{pred}$. For a chosen acceptance threshold $\eta \in (0, 1)$, a typical strategy rejects a step with $\rho < \eta$ and reduces the radius, and accepts a step with $\rho \geq \eta$. Strong agreement can justify increasing the radius. Equivalent damping strategies increase λ after poor steps and may decrease it after good steps.

These rules explain the mechanism, but are not a universal convergence theorem: a complete theorem specifies the model-accuracy assumptions, subproblem accuracy and radius-update rules. A simple rigorous alternative is to put LM directions into the Armijo framework. If $\lambda_k \in [\lambda_{\min}, \lambda_{\max}]$ with $\lambda_{\min} > 0$ and $\|J(\boldsymbol{x}_k)\|$ is uniformly bounded, then $B_k = J_k^T J_k + \lambda_k I$ has the spectral bounds required by the global stationarity result. Under that theorem's remaining assumptions, the resulting line-search method drives $\|\nabla f(\boldsymbol{x}_k)\|$ to zero.

4.12.4. The relation between damping and the local convergence rate

At a full-rank stationary point and for fixed $\lambda > 0$, the LM iteration map has derivative

$$DT_\lambda(\boldsymbol{x}_*) = (A_* + \lambda I)^{-1}(\lambda I - S_*). \tag{4.22}$$

This follows by differentiating the map as in (4.16). At a zero-residual solution, $S_* = 0$, and the eigenvalues are $\lambda/(\sigma_i(J_*)^2 + \lambda) \in (0, 1)$. Fixed positive damping therefore generally gives linear rather than quadratic local convergence.

More quantitatively, under the zero-residual hypotheses of the zero-residual convergence theorem, write $\boldsymbol{r}(\boldsymbol{x}) = J(\boldsymbol{x})\boldsymbol{e} + \boldsymbol{d}(\boldsymbol{x})$ with $\|\boldsymbol{d}(\boldsymbol{x})\| \le C\|\boldsymbol{e}\|^2$. The damped update satisfies

$$\boldsymbol{e}_+ = (A + \lambda I)^{-1}\big(\lambda \boldsymbol{e} - J^T \boldsymbol{d}(\boldsymbol{x})\big).$$

Uniform local rank bounds imply

$$\|\boldsymbol{e}_+\| \le C_1 \lambda \|\boldsymbol{e}\| + C_2 \|\boldsymbol{e}\|^2 .$$

Consequently, a locally convergent full-step scheme with $\lambda_k = O(\|\boldsymbol{e}_k\|)$ has a quadratic error bound. This is a sufficient rate condition, not a prescription that assumes access to the unknown error.

4.13. Implementation and interpretation

In an implementation, we need to distinguish several tests, since each one provides different information about the computed solution.

1. **First-order stationarity.** We monitor $\big\|J^T \boldsymbol{r}\big\|$, using scales and tolerances appropriate to the variables. A small residual is stronger in some fitting problems but need not be achievable. A small gradient alone does not certify a minimum.

2. **Step size and stagnation.** A small $\|\boldsymbol{p}\|$ can result from excessive damping, poor scaling or numerical failure. It should not be the sole success criterion.

3. **Linearized rank.** We examine singular values or use rank-revealing factorizations. A formally unique least-squares step can still be highly sensitive to perturbations.

4. **Derivative consistency.** For a chosen direction $\boldsymbol{d}$, we compare $J(\boldsymbol{x})\boldsymbol{d}$ with $[\boldsymbol{r}(\boldsymbol{x} + h\boldsymbol{d}) - \boldsymbol{r}(\boldsymbol{x})]/h$ over a range of h. Truncation and roundoff compete, so taking h arbitrarily small is ineffective.

5. **Acceptance.** We evaluate the original objective at the trial point. A small value of the linearized model does not ensure a small nonlinear objective.

The damped step can be computed without forming $J^T J$ by solving

$$\min_{\boldsymbol{p}} \left\| \begin{pmatrix} J \\ \sqrt{\lambda} I \end{pmatrix} \boldsymbol{p} + \begin{pmatrix} \boldsymbol{r} \\ 0 \end{pmatrix} \right\|^2 .$$

For large problems, factorization costs and access to Jacobian-vector products become central, but those topics require additional numerical analysis.

4.13.1. Stopping criteria and derivative checks

For a general smooth objective, we monitor a scaled gradient norm together with

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|, \qquad |f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k)|.$$

Small changes may indicate stagnation rather than stationarity. A small gradient may indicate a saddle rather than a minimum. Tolerances should reflect measurement and discretization errors, as well as the accuracy of any simulation used to evaluate the objective.

We can check a supplied gradient in several directions by comparing

$$\frac{f(\boldsymbol{x} + h\boldsymbol{d}) - f(\boldsymbol{x})}{h} \quad \text{with} \quad \nabla f(\boldsymbol{x})^T \boldsymbol{d}.$$

We need to consider a range of h: truncation dominates at large h, while subtractive cancellation dominates at very small h. Automatic differentiation differentiates the implemented computation without finite-difference subtraction, but still requires a correct model and care with nonsmooth operations. For PDE-constrained optimization, adjoint methods often compute a gradient at a cost largely independent of the number of design variables.

4.14. Which method should we choose?

Gradient descent provides a simple low-storage baseline; nonlinear CG stores curvature history in a few vectors. BFGS suits smooth problems of moderate dimension, and L-BFGS large ones. Newton and truncated Newton are attractive when reliable Hessian information and efficient linear solves are available.

For sums of squares, we exploit the residual structure. Gauss–Newton is effective near a well-conditioned, small-residual solution, but its convergence depends on residual curvature relative to $J^T J$, not on residual size alone. Levenberg–Marquardt handles damping and rank management; fast local convergence requires suitable damping reduction. General trust-region methods help with poor initial guesses and indefinite curvature.

For nonconvex problems, we scale variables, verify derivatives, and consider several starting points. Under the stated assumptions, global stationarity means that the gradient tends to zero, without guaranteeing a global minimizer. Local error rates require sufficiently close starting points. We assess results against the original modelling objective, reporting stationarity, residuals, and sensitivity; the optimizer's status message alone does not provide this information.

5. Discrete mathematics and data structures

Many models combine continuous quantities with a discrete structure: intersections joined by roads, cells sharing a boundary, or tasks with precedence constraints. We briefly recall how to represent these relations and search them efficiently. Our aim is to connect each algorithm with its assumptions and the data structure that determines its cost.

5.1. Graphs and their representation

Definition 5.1.1. A finite graph $G = (V, E)$ consists of vertices $V = \{1, \dots, n\}$ and edges E. In an undirected graph an edge is an unordered pair $\{u, v\}$; in a directed graph it is an ordered pair (u, v), permitting travel from u to v. We write $m = |E|$ and, unless stated otherwise, consider simple graphs, without self-loops or parallel edges.

A walk follows successive edges, respecting their directions when present. A path has no repeated vertices. A cycle is a closed walk of positive length with no repeated vertices apart from its equal endpoints; in a simple undirected graph we require at least three edges. A vertex is *reachable* from s if a path from s exists. An undirected graph is connected if every vertex is reachable from every other one. When an edge represents a route, we assign a cost ℓ_e and add costs along a path. In particular, unit costs measure the number of edges traversed.

The representation affects the cost of a search. An *adjacency matrix* has $A_{uv} = 1$ when u is joined to v, and zero otherwise. It uses $O(n^2)$ storage and allows constant-time adjacency queries, but finding all neighbours of one vertex requires scanning a row. An *adjacency list* stores, for each vertex, its outgoing neighbours and any associated costs. Its total storage is $O(n + m)$; an undirected edge appears in both endpoint lists. Scanning all lists takes $O(n + m)$ operations, making this representation suitable for sparse graphs. If zero edge costs are allowed, a cost matrix must distinguish a missing edge from an edge of cost zero, for example by using $+\infty$ for absence.

5.2. Trees, queues, stacks, and heaps

Definition 5.2.1. A *tree* is a connected undirected graph without cycles. A *forest* is an undirected graph whose connected components are trees.

In a tree there is exactly one path between any two vertices: two distinct paths would produce a cycle. For $n > 1$, an endpoint of a longest path is a leaf, a vertex of degree one: another neighbour would either extend the path or produce a cycle. Deleting a leaf preserves connectedness and the absence of cycles. Induction, starting from the one-vertex tree, therefore shows that a tree on n vertices has $n - 1$ edges. Choosing a root gives every other vertex a unique parent, namely the preceding vertex on its path from the root. Graph searches construct such parent relations, allowing us to recover paths without storing every path explicitly.

We also need to specify how the next vertex is selected. A *queue* removes items in the order in which they were inserted: first in, first out. A *stack* removes the most recently inserted item: last in, first out. Queues and stacks admit implementations with constant-time insertion and removal, with amortized constant time for a growing array stack. Removing the first element of a contiguous array by shifting all remaining elements would not have this cost.

A *priority queue* instead removes an item of smallest key. One implementation is a binary min-heap: a complete binary tree, filled level by level from left to right, in which every parent key is at most its children's keys. With array indices starting at one, the children of position i are $2i$ and $2i+1$, when present. The height is $O(\log n)$, so insertion and extraction of the minimum take $O(\log n)$ operations; reading the minimum takes $O(1)$. An indexed heap also records each vertex's position, permitting a decrease of its key in $O(\log n)$ time. The heap orders parents against children; it need not sort all its items.

5.3. Breadth-first search

Let us start from a source s. Breadth-first search (BFS) uses a queue to explore vertices in successive distance layers. Initially only s is discovered: we set $d(s) = 0$, mark it, and insert it into the queue. All other distances are initialized to $+\infty$. While the queue is nonempty, we remove its first vertex u and examine its outgoing neighbours. For every undiscovered neighbour v, we set

$$d(v) = d(u) + 1, \qquad \text{parent}(v) = u,$$

mark v, and insert it at the back of the queue. We mark on insertion, so each vertex enters the queue at most once.

Lemma 5.3.1. *BFS computes the minimum number of edges from s to every reachable vertex. Unreachable vertices retain distance $+\infty$.*

Proof. The queue processes all vertices assigned distance k before any vertex assigned distance $k + 1$. Every discovery therefore provides a path of the assigned length. Conversely, a vertex at true distance $k + 1$ has a predecessor at distance k on a shortest path. Induction on k shows that this predecessor is processed in layer k, discovering the vertex no later than layer $k + 1$. Its assigned distance is consequently exact. □

With adjacency lists, every reachable vertex is processed once and each of its outgoing edges is examined once. Including initialization, the cost is $O(n + m)$ time and $O(n)$ auxiliary storage. On an undirected graph, restarting from each undiscovered vertex identifies all connected components at the same total asymptotic cost. Following parent pointers back to s recovers a shortest path. Unequal edge costs require a different selection rule.

5.4. Depth-first search

Depth-first search (DFS) follows a path as far as possible before returning to an earlier branching point. In a recursive implementation, we mark a vertex u on entry, examine its neighbours, and recursively visit each still-undiscovered neighbour v, setting its parent to u. When no such neighbour remains, the call returns. The active calls form a stack; an explicit stack storing the vertex and the next neighbour to examine gives the same traversal without recursion.

As in BFS, discovery marks prevent repeated visits, and the parent edges form a search tree on the reachable vertices. Restarting from undiscovered vertices gives a search forest. With adjacency lists, the complete search costs $O(n + m)$ time and $O(n)$ auxiliary storage; a long path may require stack depth n. The traversal depends on neighbour order and does not in general minimize path length.

DFS also records when a vertex is entered and when its exploration finishes. In a directed graph, an edge to a vertex still on the active stack identifies a directed cycle. If no such edge exists, reversing the finishing order gives a *topological order*, in which every edge points from an earlier vertex to a later one. This is useful for models with dependencies. For undirected cycle detection, the edge back to a vertex's parent must be excluded from this test.

5.5. Dijkstra's algorithm

Let us now assign nonnegative costs $\ell_{uv} \geq 0$ to the edges. We seek the minimum total cost $\delta(s, v)$ from s to every vertex, with $\delta(s, v) = +\infty$ if v is unreachable. Dijkstra's algorithm [5] maintains tentative distances $d(v)$ and a set S of *settled* vertices, whose distances are final. Initially $S = \varnothing$, $d(s) = 0$, and $d(v) = +\infty$ for $v \neq s$. A min-priority queue contains the unsettled vertices, keyed by $d(v)$. At each iteration we perform the following operations:

1. Extract a vertex u with minimum tentative distance. If this distance is infinite, stop; otherwise add u to S.

2. For every edge $u \to v$ with $v \notin S$, compare $d(v)$ with $d(u) + \ell_{uv}$. If the latter is smaller, set
$$d(v) \leftarrow d(u) + \ell_{uv}, \qquad \text{parent}(v) \leftarrow u,$$
and decrease v's key in the priority queue.

The update is called *relaxation*. Every finite tentative distance is the cost of an actual path; it is therefore an upper bound on the shortest-path distance.

Lemma 5.5.1. *With nonnegative edge costs, every vertex u is settled with $d(u) = \delta(s, u)$. Upon termination the remaining vertices are unreachable.*

Proof. We argue by induction over settled vertices; the source is settled first with its correct distance zero. Suppose the next extracted vertex u had a strictly shorter path than $d(u)$. On such a shortest path, let y be the first unsettled vertex and x its settled predecessor. When x was settled, relaxation gave y a tentative distance no greater than the cost of the path prefix ending at y. Nonnegative costs make this prefix no longer than the entire path to u. Hence $d(y) < d(u)$, contradicting the minimum-key choice. Thus $d(u)$ is exact. If the minimum remaining key is infinite, a path to any unsettled vertex would similarly contain an edge from a settled vertex that supplied a finite key, which is impossible. □

An indexed binary heap performs at most n extractions and m decrease-key operations. With adjacency lists, the total cost is $O((n + m) \log n)$ time and $O(n)$ auxiliary storage. A dense implementation that finds each minimum by scanning all unsettled vertices instead costs $O(n^2)$ time. To find a path to one target, we may stop when that target is settled, but not when it is first discovered. Negative costs invalidate the argument above and require another algorithm, such as Bellman–Ford, which can also detect reachable negative cycles.

Example 5.5.2. Consider the directed edges $s \to t$, $s \to a$, and $a \to t$, with costs 8, 1, and 1, respectively. BFS chooses the direct route to t, which has one edge but cost 8. Dijkstra initially assigns $d(t) = 8$ and $d(a) = 1$, then settles a and improves $d(t)$ to 2. The minimum-cost route therefore has two edges. DFS can return either route, depending on neighbour order. The example distinguishes discovering a vertex from certifying its shortest distance.

For reachability and structural exploration, we use BFS or DFS; for minimum-hop paths, BFS suffices. For nonnegative route costs we use Dijkstra, choosing the priority queue to suit graph sparsity. The lecture notes [4] provide further details on these data structures and searches. Chapter 9 applies these tools to road graphs and develops their spectral and centrality properties.

Part II.

Case studies

6. Population models and epidemics

In this chapter we consider models describing how individuals move, reproduce, interact, or change status. The word "population" need not refer to an entire country: it may denote a species in an ecosystem, the cells in a culture, or the people in one stage of an infection. We may model different populations, as in predator–prey models, or divide a single population into compartments, as in SIR models for epidemics. In both cases, the underlying structure is the same.

Epidemic models provide a concrete example. We might first try to describe the evolution of an epidemic by tracking individual infections and recoveries. For a large country, however, this is generally unfeasible. We can instead divide the inhabitants into a few *compartments* and model the average rates of transfer between them. This gives small nonlinear systems that are easy to simulate, but already require us to address the main questions of data-driven modelling: which mechanisms to include, which variables are observed, which parameters can be recovered, and whether a fit to past data can be used to predict the future.

We shall study the classical SIR model of Kermack and McKendrick [15, 12], followed by variants that include treatment, isolation, and a latent exposed state. We will consider preserved quantities in the continuous and discrete settings, together with the expected asymptotic behaviour. For predator–prey models, we will also discuss first integrals and energies that allow us to write the equations as Hamiltonian systems.

6.1. From balance laws to population models

We start with a single population $x(t)$. Its balance law is

$$\dot{x} = \text{births} - \text{deaths} + \text{immigration} - \text{emigration}.$$

The exponential model $\dot{x} = ax$ assumes that the net contribution of one individual is independent of the population size. Since individuals are counted discretely, we interpret this equation as a continuous approximation of the population. Its right-hand side is the *rate* of population increase per unit time.

Exercise 6.1.1. Assume the right hand side is some fixed rate $\rho > 0$. How long does it take for the population to double? Does it make sense for the right hand side to be negative?

We can account for limited resources by modifying the growth law. This leads to the logistic equation

$$\dot{x} = ax\left(1 - \frac{x}{K}\right),$$

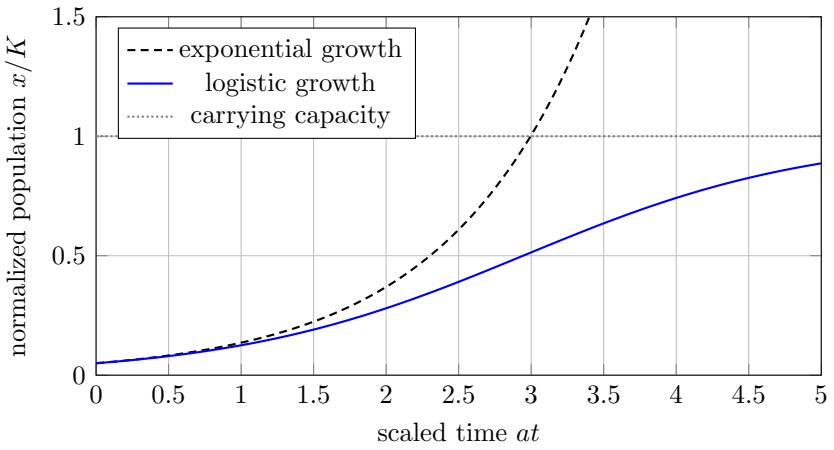


Figure 6.1.: Exponential and logistic growth from the same initial population, $x_0/K = 0.05$. Exponential growth eventually exceeds every bound, whereas the logistic trajectory approaches the carrying capacity K.

where K is the carrying capacity. This model is dissipative: for $a > 0$, every positive solution approaches K. To see this, we first observe that $x \equiv K$ is a solution of the ODE. By the Cauchy–Lipschitz theorem, other solutions cannot cross it. If $x(t) < K$, we have $\dot{x} > 0$, and the solution therefore has a limit. The only possible limit is $\lim_{t\to\infty} x(t) = K$, since all other possibilities have a positive derivative.

The contrast with unrestricted growth is especially clear after scaling the population by K and time by $1/a$. For the same initial value $x_0 > 0$, the two solutions are

$$x_{\exp}(t) = x_0 e^{at}, \qquad x_{\log}(t) = \frac{K}{1 + (K/x_0 - 1)e^{-at}}.$$

Figure 6.1 compares them for $x_0/K = 0.05$. The curves initially agree because resources are abundant, but only the logistic model reduces the per-capita growth rate as the population approaches the carrying capacity.

These observations show that we should not expect every population model to have a conserved energy. Such a quantity can nevertheless arise in models involving exchanges or interactions between populations. As an example, let us consider the Lotka–Volterra predator–prey model

$$\dot{x} = x(a - by), \qquad \dot{y} = y(-c + dx), \qquad a, b, c, d > 0.$$

Here, x is the prey population and y is the predator population. The prey grows exponentially in the absence of predators, while the predator population decays exponentially in the absence of prey. The rates $-by$ and dx describe the effect of predation on the prey and predator populations, respectively.

Lemma 6.1.2. *For positive populations, the following quantity is constant:*

$$\mathcal{E}_{\mathrm{LV}}(x, y) = dx - c\log x + by - a\log y \tag{6.1}$$

Proof. We compute

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{E}_{\mathrm{LV}} = (dx - c)(a - by) + (by - a)(-c + dx) = 0.$$

□

The logarithmic variables $q = \log x$ and $p = \log y$ turn the same system into the canonical Hamilton equations

$$\dot{q} = \frac{\partial H}{\partial p}, \qquad \dot{p} = -\frac{\partial H}{\partial q},$$

with

$$H(q, p) = ap - be^p + cq - de^q = -\mathcal{E}_{\mathrm{LV}}(e^q, e^p).$$

We often encounter these equations in mechanics, where $H(q, p)$ represents the total energy. We can verify that this energy is preserved along trajectories by computing

$$\frac{\mathrm{d}}{\mathrm{d}t}H(q, p) = \frac{\partial H}{\partial q}\dot{q} + \frac{\partial H}{\partial p}\dot{p} = \frac{\partial H}{\partial q}\frac{\partial H}{\partial p} - \frac{\partial H}{\partial p}\frac{\partial H}{\partial q} = 0.$$

For mechanical systems, the Hamiltonian is usually the sum of kinetic and potential energy. Here it is a more abstract quantity, but it illustrates a recurring pattern: multiplicative population laws become additive after taking logarithms, and a first integral describes the trajectories in the phase plane.

To recover the Hamiltonian equations explicitly, we make the change of variables $q = \log x$ and $p = \log y$ introduced above. We obtain

$$\dot{q} = a - be^p, \qquad \dot{p} = -c + de^q.$$

The variables are now separated, so choosing $H(q, p) = -\mathcal{E}_{\mathrm{LV}}(e^q, e^p)$ gives the canonical Hamilton equations.

Example 6.1.3. Let us take $a = b = c = d = 1$ and $(x(0), y(0)) = (1.5, 0.5)$. Figure 6.2 shows the resulting numerical trajectory. The predator maximum follows the prey maximum: abundant prey first supports predator growth, after which the larger predator population drives the prey down. In the phase plane the trajectory encircles the positive equilibrium $(1, 1)$ and remains on the level $\mathcal{E}_{\mathrm{LV}} = 2.28768$.

6.2. The SIR model

Let us now return to epidemic models, and let $S(t)$, $I(t)$, and $R(t)$ denote the numbers of susceptible, infectious, and removed individuals. We use the term "removed" in place of "recovered" to avoid confusion: the removed compartment may include individuals who have died or are otherwise isolated and no longer contribute to transmission. The

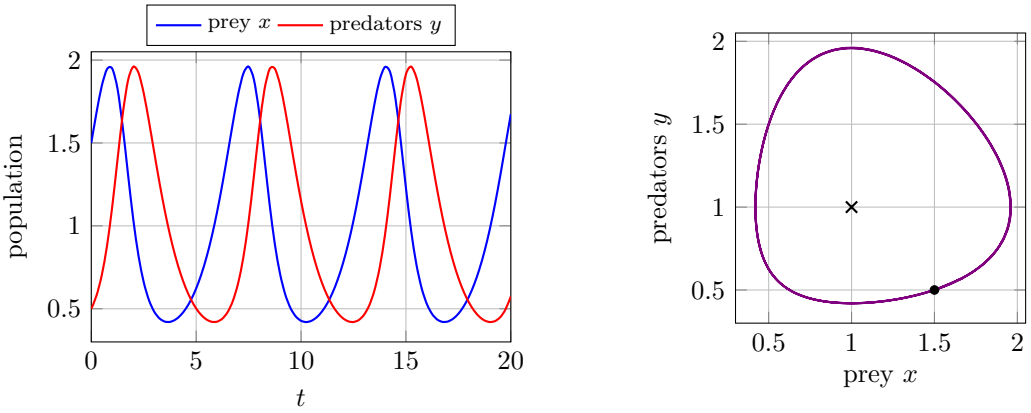


Figure 6.2.: A Lotka–Volterra solution for $a = b = c = d = 1$. Left: prey and predator populations oscillate, with the predators lagging behind the prey. Right: the same trajectory forms a closed energy level around the equilibrium $(1, 1)$; the dot marks the initial state.

key assumption is that individuals in the removed compartment do not return to the susceptible or infectious compartments. The basic model is

$$\begin{cases} \dot{S} &= -\beta \frac{SI}{N}, \\ \dot{I} &= \beta \frac{SI}{N} - \gamma I, \\ \dot{R} &= \gamma I, \end{cases} \tag{6.2}$$

We immediately verify that $S + I + R = N$ is constant, so the model is closed. Indeed, the time derivative of the total population is the sum of $\dot{S} + \dot{I} + \dot{R}$, which is zero.

The transmission rate β and removal rate γ have units of inverse time. In particular, $1/\gamma$ is the mean infectious time under the exponential waiting-time assumption.

One infectious individual makes effective contacts at rate β; the probability that a randomly selected contact is susceptible is S/N. This derivation relies on the following assumptions:

- the population is closed during the modeled interval: the total population $S+I+R$ is constant;
- individuals mix homogeneously and independently;
- the parameters do not change in time;
- removal gives immunity, or at least lasts longer than the time interval under consideration;
- a large-population deterministic average is adequate.

We remark that each assumption is a possible source of model error. A model may therefore be adequate in one setting and require adjustment in another. We can start with the simplest model and check whether it captures the relevant features of the system, adding information when needed. Increasing complexity also introduces the risk of overfitting: with enough parameters, we may describe the observations without obtaining an interpretable model or reliable predictions of future evolution.

It is often convenient to work with the fraction of the population in each compartment. We set

$$s = \frac{S}{N}, \qquad i = \frac{I}{N}, \qquad r = \frac{R}{N}.$$

Then

$$\dot{s} = -\beta si, \qquad \dot{i} = \beta si - \gamma i, \qquad \dot{r} = \gamma i, \qquad s + i + r = 1. \tag{6.3}$$

The simplex

$$\{(s, i, r) : s, i, r \geq 0,\ s + i + r = 1\}$$

is invariant: every solution with initial data in this simplex remains there for all positive times. For example,

$$s(t) = s(0) \exp\left(-\beta \int_0^t i(u)\,\mathrm{d}u\right),$$

so a positive susceptible population cannot cross zero. Similar formulas, or a boundary argument, establish positivity of the other compartments.

Definition 6.2.1. The *basic reproduction number* $\mathcal{R}_0$ is the expected number of secondary infections generated by one infectious individual over its infectious lifetime, when introduced into an otherwise fully susceptible population. It is a dimensionless quantity.

To compute this quantity for SIR, let us consider a stage when $s \approx 1$, so that almost the entire population is susceptible and exposed to the disease. We first ignore the new infections produced by the initial infectious group and follow only the removal of its original members. If its initial size is I_0, then its size $J(t)$ satisfies

$$\dot{J} = -\gamma J, \qquad J(0) = I_0,$$

and therefore

$$J(t) = I_0 e^{-\gamma t}.$$

Thus the original infectious group persists over time with the exponentially decaying profile $e^{-\gamma t}$.

When the population is fully susceptible, each infectious individual generates new infections at rate β. Hence the original group generates secondary infections at time t at rate

$$\beta J(t) = \beta I_0 e^{-\gamma t}.$$

The total number of secondary infections generated by this group over its entire infectious lifetime is therefore

$$\int_0^\infty \beta I_0 e^{-\gamma t}\,\mathrm{d}t = \beta I_0 \int_0^\infty e^{-\gamma t}\,\mathrm{d}t = \frac{\beta}{\gamma} I_0.$$

Dividing by the size I_0 of the initially infectious group gives the number of secondary infections generated per initially infectious individual:

$$\mathcal{R}_0 = \frac{\beta}{\gamma}.$$

The same quantity appears naturally in the dynamics near the fully susceptible disease-free state $(s, i, r) = (1, 0, 0)$. Since

$$\dot{i} = (\beta s - \gamma) i,$$

for $s \approx 1$ we obtain

$$\dot{i} \approx (\beta - \gamma) i = \gamma(\mathcal{R}_0 - 1) i.$$

Consequently, a small introduction of infection initially grows if $\mathcal{R}_0 > 1$ and decays if $\mathcal{R}_0 < 1$. Hence, $\mathcal{R}_0$ indicates whether the epidemic is initially expanding or declining. We should distinguish this threshold from the initial exponential growth rate. If the initial susceptible fraction is s_0, freezing s at its initial value gives

$$i(t) \approx i_0 \exp\big((\beta s_0 - \gamma) t\big).$$

The rate $\beta s_0 - \gamma$ has units of inverse time. Multiplying both β and γ by the same positive constant leaves $\mathcal{R}_0$ unchanged but changes the time scale of the epidemic. More generally, when only a fraction $s(t)$ of the population remains susceptible, the corresponding effective reproduction number is

$$\mathcal{R}_{\text{eff}}(t) = \mathcal{R}_0 s(t) = \frac{\beta s(t)}{\gamma}.$$

Indeed,

$$\dot{i} = \gamma i \big(\mathcal{R}_{\text{eff}}(t) - 1\big),$$

so, for $i(t) > 0$, infections increase when $\mathcal{R}_{\text{eff}}(t) > 1$ and decrease when $\mathcal{R}_{\text{eff}}(t) < 1$.

If $i_0 > 0$ and $i(t)$ has an interior maximum at $t = t_*$, then $\dot{i}(t_*) = 0$, and hence

$$s(t_*) = \frac{\gamma}{\beta} = \frac{1}{\mathcal{R}_0}.$$

6.2.1. A first integral and the final size

The threshold describes the initial direction of the motion. To study its subsequent evolution, we use a quantity that remains constant along the entire trajectory.

Proposition 6.2.2. *Let $\beta, \gamma > 0$ and let (s_0, i_0, r_0) belong to the population simplex, with $s_0 > 0$. Along the solution of* (6.3)*,*

$$\mathcal{E}(s, i) = s + i - \frac{\gamma}{\beta} \log s \tag{6.4}$$

is constant. Moreover, there exist limits $s_\infty > 0$ and $r_\infty = 1 - s_\infty$ such that

$$s(t) \longrightarrow s_\infty, \qquad i(t) \longrightarrow 0, \qquad r(t) \longrightarrow r_\infty.$$

The limiting susceptible fraction satisfies

$$\log\left(\frac{s_\infty}{s_0}\right) = -\mathcal{R}_0(r_\infty - r_0) = -\mathcal{R}_0(s_0 + i_0 - s_\infty). \tag{6.5}$$

Proof. Positivity of s allows us to differentiate (6.4):

$$\frac{\mathrm{d}\mathcal{E}}{\mathrm{d}t} = \dot{s} + \dot{i} - \frac{\gamma}{\beta}\frac{\dot{s}}{s} = -\gamma i + \gamma i = 0.$$

Since s is nonincreasing and r is nondecreasing, and both belong to $[0, 1]$, they have limits. Hence $i = 1 - s - r$ also has a limit. Integrating $\dot{r} = \gamma i$ gives

$$\int_0^\infty i(t)\,\mathrm{d}t = \frac{r_\infty - r_0}{\gamma} < \infty,$$

so the nonnegative limit of i must be zero. Finally, integrating $\dot{s}/s = -\beta i$ yields

$$s(t) = s_0 \exp\left(-\frac{\beta}{\gamma}(r(t) - r_0)\right) \geq s_0 \exp\left(-\frac{\beta}{\gamma}(1 - r_0)\right) > 0.$$

Taking the limit proves both positivity of s_∞ and (6.5). □

When $r_0 = 0$, the final-size equation becomes $\log(s_\infty/s_0) = -\mathcal{R}_0(1 - s_\infty)$. It determines the final size implicitly. The first integral also determines the height of the infectious peak. If $i_0 > 0$ and $\mathcal{R}_0 s_0 > 1$, the susceptible fraction decreases through $1/\mathcal{R}_0$; otherwise i could not tend to zero. Substitution in (6.4) gives

$$i_{\max} = s_0 + i_0 - \frac{1}{\mathcal{R}_0} - \frac{1}{\mathcal{R}_0}\log(\mathcal{R}_0 s_0).$$

If $\mathcal{R}_0 s_0 \leq 1$, i is nonincreasing and its maximum is i_0.

6.3. Adding detection and treatment: an SIRT model

We can extend SIR to include detection and treatment. In the SIRT model, T denotes detected or treated individuals who have been traced and effectively isolated, and therefore no longer transmit. Using lower-case letters for the other population fractions, we write $z = T/N$ to distinguish the treated fraction from the time variable t. We consider

$$\begin{aligned} \dot{s} &= -\beta s i, \\ \dot{i} &= \beta s i - (\gamma + \tau) i, \\ \dot{z} &= \tau i - \delta z, \\ \dot{r} &= \gamma i + \delta z. \end{aligned} \tag{6.6}$$

Here τ is the detection-and-isolation rate, γ is direct removal from the undetected infectious class, and $1/\delta$ is the mean time spent in treatment or isolation. Adding the equations shows that $s + i + z + r = 1$ is conserved.

The active subsystem (s, i) is again an SIR system, with total exit rate $\alpha = \gamma + \tau$ obtained by combining the T and R compartments. The preceding derivation therefore gives the reproduction number and first integral

$$\mathcal{R}_0 = \frac{\beta}{\gamma + \tau}, \qquad \mathcal{E}_T(s, i) = s + i - \frac{\gamma + \tau}{\beta} \log s.$$

In particular, $\mathcal{E}_T$ is constant along trajectories with $s > 0$. Once s and i have been computed, the equations for z and r are linear forced equations.

The additional compartment also allows us to distinguish the state from its measurement. If every transition into T is reported, then the expected number of new reports during day k is

$$\widehat{y}_k(\boldsymbol{\theta}) = N \int_{t_{k-1}}^{t_k} \tau i(t; \boldsymbol{\theta}) \, \mathrm{d}t, \tag{6.7}$$

and $Nz(t_k)$ approximates the number currently isolated. This observation model is usually more defensible than identifying all reported cases with the hidden infectious state $Ni(t_k)$.

The energy above relies on complete isolation. If treated people retain a relative infectivity $\eta > 0$, the incidence becomes $\beta s(i + \eta z)$, the (s, i) subsystem no longer closes, and the simple $\mathcal{E}_T$ is generally lost. Conservation of total population remains.

6.4. Tracking exposure before infectiousness

For many infections there is a latent interval after acquisition of the pathogen but before the individual becomes infectious. The SEIR model introduces an exposed compartment E:

$$\begin{aligned} \dot{s} &= -\beta s i, \\ \dot{e} &= \beta s i - \sigma e, \\ \dot{i} &= \sigma e - \gamma i, \\ \dot{r} &= \gamma i. \end{aligned} \tag{6.8}$$

The exponential-stage interpretation assigns mean latent time $1/\sigma$ and mean infectious time $1/\gamma$. Under the assumptions of (6.8), the basic reproduction number remains β/γ: the latent stage delays transmission but does not remove individuals before they become infectious.

Let us assume $\beta, \sigma, \gamma > 0$, nonnegative initial fractions with sum one, and $s_0 > 0$. The corresponding first integral is

$$\mathcal{E}_E(s, e, i) = s + e + i - \frac{\gamma}{\beta} \log s,$$

as differentiation shows. As in SIR, s and r have limits, and $\int_0^\infty i(t)\,\mathrm{d}t < \infty$. Integrating $\dot{i} = \sigma e - \gamma i$ also bounds $\int_0^\infty e(t)\,\mathrm{d}t$. Since the derivatives of both compartments are bounded on the invariant simplex, e and i are uniformly continuous. Their nonnegativity and finite integrals then imply $e(t), i(t) \to 0$. The same integration of $\dot{s}/s$ gives

$$\log\left(\frac{s_\infty}{s_0}\right) = -\mathcal{R}_0(r_\infty - r_0) = -\mathcal{R}_0(s_0 + e_0 + i_0 - s_\infty).$$

Thus, for fixed β, γ, s_0, r_0, changing the positive latent rate σ changes the timing and shape of the wave but leaves the final-size equation unchanged. Comparison with SIR requires matching s_0 and r_0, so that its initial infectious fraction equals $e_0 + i_0$. Final totals alone cannot identify σ; time-resolved data are essential.

6.4.1. Other useful variations

We introduce additional compartments only when the data and the modelling question justify them. Common extensions include:

- an SIRS model, with a flow from R back to S to represent waning immunity;
- births, background deaths, and migration for endemic dynamics;
- age, location, or contact-network classes to replace homogeneous mixing;
- separate hospitalized and deceased compartments;
- a time-dependent transmission rate $\beta(t)$ to represent behavior, policy, seasonality, or a changing variant.

More detail can reduce structural bias but increases the number of hidden states and frequently worsens parameter identifiability.

The last variation is sometimes called SIR-T, where T denotes time rather than a fourth population.

6.5. Implementing the forward ODE solver

Our aim is to construct an epidemic model from observed data. Before estimating its parameters, we need a forward solver that is reliable over the entire admissible parameter set. Its error need not be close to machine precision, but it must be small enough that changing the solver tolerance does not materially change the observables or the objective function. Otherwise the optimizer is partly fitting time-discretization error.

We refer to Chapter 2 for an overview of ODE integration methods. In general, we use implicit integrators for stiff systems and explicit Runge–Kutta methods for smooth nonstiff problems. To assess stiffness, we look at the Jacobian of the system of ODEs, which is

$$J(S,I,R) = \begin{bmatrix} -\beta I/N & -\beta S/N & 0 \\ \beta I/N & \beta S/N - \gamma & 0 \\ 0 & \gamma & 0 \end{bmatrix}.$$

A norm bound alone does not determine whether the problem is stiff. We must compare the step size allowed by explicit stability for rapidly decaying modes with the step size needed for accuracy on the time scales of interest. We return to this comparison below when choosing the solver. Notice also that, since N is constant and all three compartments are scaled by the same factor, the count and fraction Jacobians agree at corresponding states.

6.5.1. A vector-valued implementation

We first write the system of ODEs as a single vector equation, storing the state in a fixed, documented order. For the SIR equations in fractions, a minimal implementation using a standard initial-value solver is:

```
def sir_rhs(t, y, beta, gamma):
    s, i, r = y
    incidence = beta*s*i
    return np.array([-incidence,
                     incidence - gamma*i,
                     gamma*i])

sol = solve_ivp(sir_rhs, (0.0, 160.0), y0,
                args=(beta, gamma), method="RK45",
                rtol=1e-6, atol=1e-9,
                dense_output=True)
if not sol.success:
    raise RuntimeError(sol.message)
Y_report = sol.sol(t_report)
```

Here we use SciPy's `solve_ivp`, selecting an embedded Runge–Kutta 4/5 method. We will discuss this choice later in the section.

The solver advances the coupled vector, not three independent scalar equations. Keeping all flows in one right-hand side makes cancellations such as $\dot{s} + \dot{i} + \dot{r} = 0$ visible in the

code. Working with fractions also gives all components comparable scales. If counts are used instead, component-dependent absolute tolerances may be appropriate. Adaptive codes typically test a norm of terms of the form

$$\frac{e_j}{\mathrm{atol}_j + \mathrm{rtol}\,\max\{|y_{n,j}|, |y_{n+1,j}|\}},$$

where e_j is an estimate of the local error. Thus `rtol` controls components away from zero, while `atol` sets the accuracy scale near zero.

We need to distinguish two uses of the word *vectorized*. The function above is vector-valued: it maps one state in $\mathbb{R}^3$ to one derivative in $\mathbb{R}^3$. Some libraries also allow the right-hand side to process several state columns in one call. For SciPy's convention, the same SIR calculation can be written as

```
def sir_rhs_vectorized(t, Y, beta, gamma):
    s, i, r = Y[0], Y[1], Y[2]
    incidence = beta*s*i
    return np.stack((-incidence,
                     incidence - gamma*i,
                     gamma*i), axis=0)
```

It returns an array with the same shape as Y, whether Y has shape `(3,)` or `(3,k)`. Passing `vectorized=True` can accelerate the finite-difference Jacobians used by stiff solvers, especially for a larger system. It does not ask `solve_ivp` to integrate k independent parameter values: there is still only one initial-value problem. An ensemble may instead be run as separate solves, or stacked into a larger state when sharing one adaptive time mesh is acceptable. In the latter case, the most difficult trajectory controls the step size for the entire batch.

We also need to implement observation operators with the same care as the ODE itself. The requested reporting times are normally output points, evaluated using the solver's continuous interpolant; they need not be internal step endpoints. For an integrated flow such as (6.7), we can append the auxiliary counter

$$\dot{d} = \tau i, \qquad d(0) = 0, \qquad \widehat{y}_k = N\{d(t_k) - d(t_{k-1})\}.$$

The variable d is not another population compartment and is excluded from the population balance.

After each trial solve, we can compute the inexpensive diagnostics

$$\epsilon_{\mathrm{mass}} = \max_n |s_n + i_n + r_n - 1|, \qquad y_{\min} = \min_{n,j} y_{n,j}.$$

These tests detect coding and stability failures, but they do not by themselves establish accuracy: explicit Euler, for example, preserves the linear population balance exactly while it may approximate the individual compartments poorly. We should not simply clip small negative values, since this changes the balance law and makes the parameter-to-solution map nonsmooth. We first reduce the step or tolerance, or use a method designed to preserve positivity.

Remark 6.5.1. Runge–Kutta methods preserve linear invariants. We can therefore check that $s + i + r = 1$ at all times, up to errors of the order of machine precision.

6.5.2. Choosing the integrator

To choose an integration method, we consider stiffness and any energy or first integrals that need to be preserved. The dimension is not a limiting factor here, since the number of equations is small. It becomes relevant when the number of equations exceeds thousands.

A suitable first choice is an explicit Runge–Kutta method of medium order with adaptive step size. In fact, an explicit Runge–Kutta method only evaluates the right-hand side and is usually the most economical choice when accuracy, rather than stability, determines its step size. An implicit method such as Radau or BDF solves linear or nonlinear systems at each step, but can remain stable in the presence of rapidly decaying modes.

To assess stiffness, we examine the rapidly decaying modes associated with Jacobian eigenvalues whose real parts are negative and large in magnitude relative to the inverse time scales of interest. The rate constants provide a useful first indication. For the SIR system in fractions, the Jacobian is

$$J(s,i,r) = \begin{pmatrix} -\beta i & -\beta s & 0 \\ \beta i & \beta s - \gamma & 0 \\ 0 & \gamma & 0 \end{pmatrix}.$$

Large separations among the relevant eigenvalue magnitudes indicate different time scales. If an explicit stability restriction $h|\lambda_{\text{fast}}| \lesssim C$ is much tighter than the step needed to resolve the observable slow motion, the problem is stiff. A basic SIR model with rates of comparable size is normally nonstiff. An SEIR or SIRT model can become stiff when a latent, treatment, or reporting transition is orders of magnitude faster than epidemic growth and removal. Age-structured models and models coupled to spatial diffusion provide further examples.

This diagnosis can change along a nonlinear trajectory and as the parameters change during optimization. In practice, we can start a small compartment model with an adaptive explicit method and examine its accepted steps, rejected steps, and right-hand-side evaluations. If it takes unexpectedly small steps through a smooth, slowly changing part of the solution, we repeat this preliminary solve with Radau or BDF. A large reduction in steps or function evaluations is evidence for stiffness. When an implicit method is selected, an analytic Jacobian, or at least its sparsity pattern for a large model, can greatly reduce the cost of the internal solves. These nonlinear solves typically use Newton-type iterations; without an analytic Jacobian, the solver may approximate it by finite differences.

We also need to choose between constant and adaptive step sizes. A constant step size is useful when constructing a method from scratch, checking its order, aligning with known discontinuities, or using a specialized structure-preserving scheme. It also makes the computational work predictable. Its weakness is that one value of h must cope with both the rapid outbreak transient and the slowly varying tail.

Adaptive methods attempt to keep the local error comparable to a requested scale, shortening the step near peaks and fast transitions and lengthening it on smooth inter-

vals. This is usually preferable for a general-purpose forward model. Variable internal steps do not imply irregular output: dense output can still return values exactly at daily observation times. If a coefficient such as $\beta(t)$ jumps at a known intervention time, the integration should be split there, or the solver should at least be forced to end a step at the discontinuity; a smooth interpolant should not straddle a jump.

We can often anticipate the answer but cannot determine the best mesh fully a priori. Known rate ratios, forcing periods, discontinuities, and a rough Jacobian spectrum provide useful estimates. The actual error also depends on the trajectory, initial data, requested tolerance, and—during parameter estimation—the trial parameters. A short pilot run is therefore part of method selection. A trustworthy final calculation is insensitive in its quantities of interest to a smaller fixed step or tighter adaptive tolerance.

6.5.3. Two simplified numerical experiments

Example 6.5.2 (A nonstiff SIR problem). Let us first consider (6.3) with

$$\beta = 0.3 \text{ day}^{-1}, \qquad \gamma = 0.1 \text{ day}^{-1}, \qquad (s_0, i_0, r_0) = (0.999, 0.001, 0), \qquad 0 \le t \le 160.$$

The reference trajectory has its infection peak at $t = 38.3553$ days, where $i = 0.300796$. Table 6.1 compares simple fixed-step implementations with adaptive RK45. Errors are the largest componentwise differences at the integer days from a tight DOP853 reference calculation.

method	error control	accepted steps	RHS evaluations	maximum error
explicit Euler	fixed $h = 1$ day	160	160	$9.69 \cdot 10^{-2}$
classical RK4	fixed $h = 1$ day	160	640	$9.26 \cdot 10^{-6}$
classical RK4	fixed $h = 0.5$ day	320	1280	$6.24 \cdot 10^{-7}$
RK45	rtol $= 10^{-6}$, atol $= 10^{-9}$	59	374	$7.07 \cdot 10^{-7}$

Table 6.1.: Fixed- and variable-step solutions of a nonstiff SIR example. The adaptive step ranged from 0.0415 to 5.29 days. Counts of accepted steps and function evaluations are solver-implementation dependent, but the accuracy comparison is representative.

At approximately the same error, adaptive RK45 uses fewer than one third of the right-hand-side evaluations required by RK4 with $h = 0.5$. Euler retains $s + i + r = 1$ to roundoff in this experiment, despite its large error and shifted epidemic peak. This illustrates why both invariants and tolerance-refinement tests are needed.

Example 6.5.3 (A stiff progression model). We now consider the linear progression

$$\dot{e} = -\sigma e, \qquad \dot{i} = \sigma e - \gamma i, \qquad \dot{r} = \gamma i, \qquad (e_0, i_0, r_0) = (1, 0, 0),$$

with $\sigma = 1000$, $\gamma = 1$, and $0 \le t \le 10$. The exact solution is

$$e(t) = e^{-1000t}, \qquad i(t) = \frac{1000}{999}\left(e^{-t} - e^{-1000t}\right), \qquad r(t) = 1 - e(t) - i(t).$$

The fast exposed mode disappears on a time scale 10^{-3}, while the infectious population changes on a time scale of order one. Explicit Euler requires $h < 2/1000$ for strict decay of the fast mode, hence more than 5000 steps merely for stability on this interval. At the boundary $h = 0.002$ the numerical exposed component alternates sign without decaying.

method	accepted steps	RHS evaluations	LU factorizations	maximum error
RK45 (explicit)	3066	21398	0	$6.5 \cdot 10^{-8}$
Radau (implicit)	192	1361	64	$7.0 \cdot 10^{-8}$

Table 6.2.: Adaptive solvers on the stiff progression model, both with rtol $= 10^{-6}$ and atol $= 10^{-9}$. The maximum error is measured against the exact solution at 10001 equally spaced output times. No analytic Jacobian was supplied.

The explicit solver remains stable and accurate, but only by taking thousands of short steps after the physically interesting fast transient has ended. Radau pays for matrix factorizations, yet takes far fewer steps and right-hand-side evaluations. Conversely, on the nonstiff SIR example those implicit solves would usually be unnecessary overhead. These experiments show how we can choose a step size according to accuracy when possible, and use an implicit method when stability would otherwise force an explicit scheme to take much smaller steps.

6.6. Parameter estimation as an inverse problem

Let $\boldsymbol{x}(t;\boldsymbol{\theta})$ be the solution of a chosen compartment model, where $\boldsymbol{\theta}$ contains rates and, possibly, unknown initial conditions. In practice, we often cannot observe $\boldsymbol{x}$ directly. Instead, a surveillance system produces measurements of the form

$$\boldsymbol{y}_k = \boldsymbol{h}_k(\boldsymbol{x}, \boldsymbol{\theta}) + \boldsymbol{\varepsilon}_k,$$

where $\boldsymbol{h}_k$ is an observation operator. It may select a compartment, integrate a transition such as (6.7), include a reporting fraction, or add a delay between infection and report. For instance, we may assume that the number of removed individuals can be measured, together with the number of susceptible individuals, possibly with some delay.

To estimate the parameters, we formulate the weighted nonlinear least-squares inverse problem

$$\min_{\boldsymbol{\theta} \in \Theta} \frac{1}{2} \sum_{k,\ell} \frac{\left(h_{k,\ell}(\boldsymbol{x}(\boldsymbol{\theta})) - y_{k,\ell}\right)^2}{\sigma_{k,\ell}^2} + \frac{\lambda}{2} \|L(\boldsymbol{\theta} - \boldsymbol{\theta}_{\mathrm{prior}})\|_2^2. \tag{6.9}$$

Every evaluation of the objective requires an ODE solve. The Gauss–Newton and Levenberg–Marquardt methods discussed earlier are therefore natural choices. Positivity can be enforced with bounds or by optimizing logarithmic parameters. For counts with variance that grows with their mean, a Poisson or negative-binomial likelihood is often more appropriate than unweighted least squares.

Before interpreting the result of an optimization, we need to examine whether its parameters are identifiable. Early in an SIR outbreak $s \simeq s_0$, and the data primarily

reveal the growth rate $\beta s_0 - \gamma$; many pairs (β, γ) produce almost the same curve. Removal observations and timestamps help separate them. In an SEIR model, the unknown initial exposed population may be confounded with σ. In SIRT, incidence alone may identify $\gamma + \tau$ but not its two parts. We remark that a small residual does not imply that a poorly conditioned combination of parameters has been accurately determined. Profile likelihoods, bootstrap resampling, the singular values of the residual Jacobian, and repeated fits over nearby time windows are useful diagnostics.

6.7. COVID-19 in South Korea

We now consider a small historical experiment, following the complete calculation from public data to verification on future observations. The purpose is not to reconstruct every mechanism of COVID-19. In particular, we would need to estimate the parameters again for a different setting, or for South Korea at a different time.

6.7.1. Data and observation model

We use the archived Johns Hopkins CSSE time series [14, 6]. The repository identifies the South-Korean source as the national Ministry of Health and Welfare. To make the calculation reproducible, we use repository commit `4360e502`, select the row `Korea, South`, and read the confirmed, recovered, and death files. The dashboard records dates of report rather than dates of infection or symptom onset, an important distinction for interpreting the parameters.

For day t_k, let

$$C_k = \text{cumulative confirmed cases}, \qquad M_k = \text{reported recoveries} + \text{reported deaths},$$

and form the reported-active proxy

$$I_k = C_k - M_k.$$

For this classroom model we set $R_k = M_k$ and $S_k = N - C_k$, with the 2020 population

$$N = 51\,836\,239$$

from the World Bank [25]. We use this as an observation convention. It does not imply that every unresolved confirmed case was infectious, or that every infection was confirmed.

The calibration interval is 15–29 March 2020, containing 15 daily observations. The subsequent 14 days, 30 March–12 April, are hidden from the inversion and kept as a future control set. At the initial date,

$$C_0 = 8162, \qquad M_0 = 510 + 75 = 585, \qquad I_0 = 7577, \qquad S_0 = N - C_0.$$

These initial values and N are fixed rather than fitted.

6.7.2. Forward solver and nonlinear least squares

We describe the calculation in sufficient detail to reproduce it and to make explicit the operations involved in solving the inverse problem. The calculation reported here was verified with Python 3.14.4, NumPy 2.5.2, and SciPy 1.18.1 [23], in double precision. Time is measured in days and the state vector is

$$\boldsymbol{y} = (S, I, R)^T, \qquad \boldsymbol{y}(0) = (51\,828\,077, 7577, 585)^T.$$

No data are smoothed or interpolated, and no state or rate is fitted at the initial time.

We optimize the logarithmic variables

$$\boldsymbol{u} = (u_1, u_2)^T = (\log\beta, \log\gamma)^T, \qquad \beta = e^{u_1}, \quad \gamma = e^{u_2}.$$

Thus every trial rate is positive although the optimizer itself is unconstrained. For a trial $\boldsymbol{u}$, the right-hand side passed to the ODE solver is

$$\boldsymbol{F}(\boldsymbol{y}; \boldsymbol{u}) = \begin{pmatrix} -q \\ q - e^{u_2} I \\ e^{u_2} I \end{pmatrix}, \qquad q = e^{u_1} \frac{SI}{N}. \tag{6.10}$$

Let us first examine the Runge–Kutta calculation. Each evaluation of the residual requires a new solution of (6.10) on $[0, 14]$. We use SciPy's `solve_ivp` with `method=RK45`, the explicit Dormand–Prince embedded pair of orders five and four [7]. If h_n is the current internal step and $\boldsymbol{k}_j$ are the shared Runge–Kutta stages, the pair produces two end-of-step approximations

$$\boldsymbol{y}_{n+1}^{[5]} = \boldsymbol{y}_n + h_n \sum_j b_j \boldsymbol{k}_j, \qquad \boldsymbol{y}_{n+1}^{[4]} = \boldsymbol{y}_n + h_n \sum_j \widehat{b}_j \boldsymbol{k}_j.$$

Their difference estimates the local error without a second complete step. In the implementation, the scaled error indicator is

$$\eta_n = \left\{ \frac{1}{3} \sum_{\ell=1}^{3} \left[\frac{y_{n+1,\ell}^{[5]} - y_{n+1,\ell}^{[4]}}{10^{-8} + 10^{-10} \max\{|y_{n,\ell}|, |y_{n+1,\ell}^{[5]}|\}} \right]^2 \right\}^{1/2}. \tag{6.11}$$

A step is accepted if $\eta_n < 1$, and the fifth-order value becomes the new state. The proposed step multiplier is

$$\min\{10, 0.9\eta_n^{-1/5}\}$$

after an accepted step and

$$\max\{0.2, 0.9\eta_n^{-1/5}\}$$

after a rejected step; following a rejection the next multiplier is also capped at one. Hence the solver chooses its own nonuniform mesh. The reporting days $t_j = j$ supplied through `t_eval` are only output points: they do not force one-day steps or restart the

solver. A quartic continuous extension of the accepted Runge–Kutta step is evaluated whenever an output day lies inside it.

The complete forward-solver settings are

$$\texttt{rtol} = 10^{-10}, \qquad \texttt{atol} = 10^{-8}, \qquad \texttt{first_step=None}, \qquad \texttt{max_step} = +\infty,$$

with no events, no user-supplied Jacobian, and no vectorization. The first step is therefore estimated automatically, while all later steps follow the controller above. These are local-error tolerances, not bounds on global error or on parameter uncertainty. At the final parameters the training solve takes 18 accepted steps and 110 right-hand-side evaluations; the adaptive mesh is independent of the 15-point reporting grid.

For each $\boldsymbol{\theta} = (\beta, \gamma)$ we integrate the count form (6.2), set

$$C_{\boldsymbol{\theta}}(t) = I_{\boldsymbol{\theta}}(t) + R_{\boldsymbol{\theta}}(t) = N - S_{\boldsymbol{\theta}}(t),$$

and minimize

$$J(\beta, \gamma) = \sum_{j=0}^{14} \left[\big(C_{\boldsymbol{\theta}}(t_j) - C_j\big)^2 + \big(I_{\boldsymbol{\theta}}(t_j) - I_j\big)^2 \right]. \tag{6.12}$$

Equivalently, the code returns the 30-component residual vector

$$\boldsymbol{r}(\boldsymbol{u}) = \begin{pmatrix} C_{\boldsymbol{u}}(t_0) - C_0 \\ \vdots \\ C_{\boldsymbol{u}}(t_{14}) - C_{14} \\ I_{\boldsymbol{u}}(t_0) - I_0 \\ \vdots \\ I_{\boldsymbol{u}}(t_{14}) - I_{14} \end{pmatrix}. \tag{6.13}$$

The two residuals at t_0 vanish identically but are retained to keep the code and notation simple. All remaining residuals are raw counts in people: the C and I series and all days receive equal weight, with no normalization, robust loss, or regularization. SciPy minimizes

$$\Phi(\boldsymbol{u}) = \frac{1}{2} \|\boldsymbol{r}(\boldsymbol{u})\|_2^2,$$

so $J = 2\Phi$ and the minimizer is exactly the one in (6.12).

We now turn to the optimization algorithm. As in the accompanying notebook, we use the Levenberg–Marquardt method implemented by `scipy.optimize.least_squares`, through MINPACK. At iterate $\boldsymbol{u}_m$, forward finite differences of the entire ODE-to-observation map construct the dense 30×2 Jacobian:

$$(G_m)_{:,k} \simeq \frac{\boldsymbol{r}(\boldsymbol{u}_m + h_k \boldsymbol{e}_k) - \boldsymbol{r}(\boldsymbol{u}_m)}{h_k}, \qquad |h_k| \simeq \sqrt{\epsilon_{\text{mach}}}\, \max\{1, |u_{m,k}|\}.$$

Each additional residual evaluation requires an ODE solution. The Jacobian being approximated is that of the complete residual vector, not the Jacobian of the SIR right-hand side.

The local Gauss–Newton model is

$$m_m(\boldsymbol{p}) = \frac{1}{2}\|\boldsymbol{r}_m + G_m\boldsymbol{p}\|_2^2.$$

Levenberg–Marquardt controls the step through a trust region. With unit parameter scaling, its step can be characterized by

$$(G_m^T G_m + \lambda_m I)\boldsymbol{p}_m = -G_m^T \boldsymbol{r}_m, \qquad \lambda_m \geq 0, \tag{6.14}$$

where the damping parameter is chosen to respect the current trust-region radius. Increasing λ_m reduces the step when the linearized residual model is unreliable; when damping is unnecessary, we recover the Gauss–Newton step. The implementation uses factorizations of the Jacobian to compute the step, rather than explicitly forming these normal equations. Agreement between the predicted and actual cost reductions determines acceptance and adjustment of the trust region.

The starting point and complete optimizer settings are

$$\begin{aligned}
(\beta^{(0)}, \gamma^{(0)}) &= (0.02, 0.05)\ \text{day}^{-1}, \\
\boldsymbol{u}_0 &= (\log 0.02, \log 0.05)^T, \\
\texttt{method} = \texttt{lm}, \quad \texttt{jac} &= \texttt{2-point}, \quad \texttt{x_scale} = 1, \\
\texttt{loss} = \texttt{linear}, \quad \texttt{ftol} &= \texttt{xtol} = \texttt{gtol} = 10^{-13}, \\
\texttt{max_nfev} &= 5000.
\end{aligned}$$

There are no explicit bounds on the log parameters. The stopping tests monitor relative cost reduction, the trust-region radius relative to the scaled parameter norm, and the alignment of the residual vector with the Jacobian columns. For unit scaling, the step-size stopping test is

$$\Delta_m < \texttt{xtol}\|\boldsymbol{u}_m\|_2.$$

These tolerances specify algorithmic stopping criteria; they do not represent numbers of correct digits.

In the verified run, the optimizer reports six residual evaluations, excluding calls used for numerical differentiation. Direct counting of calls to the residual function gives 18 ODE solutions and 1944 right-hand-side evaluations. For numerical differentiation with `lm`, the returned `njev` field is `None`; it does not provide a Jacobian evaluation count. Evaluation-count conventions can depend on the SciPy version, so we distinguish the reported count from the measured computational work. The run terminates by the relative-cost test, rather than by exhaustion of the evaluation budget. The final cost is

$$\Phi(\widehat{\boldsymbol{u}}) = 1.0524645453 \cdot 10^6, \qquad J(\widehat{\beta}, \widehat{\gamma}) = 2.1049290907 \cdot 10^6\ \text{people}^2.$$

The singular values of the final residual Jacobian in log coordinates are 9075.2 and 2973.0. Their ratio, about 3.05, shows that this particular two-parameter numerical problem is not nearly rank deficient, although it does not remove the modeling and observation uncertainties discussed below.

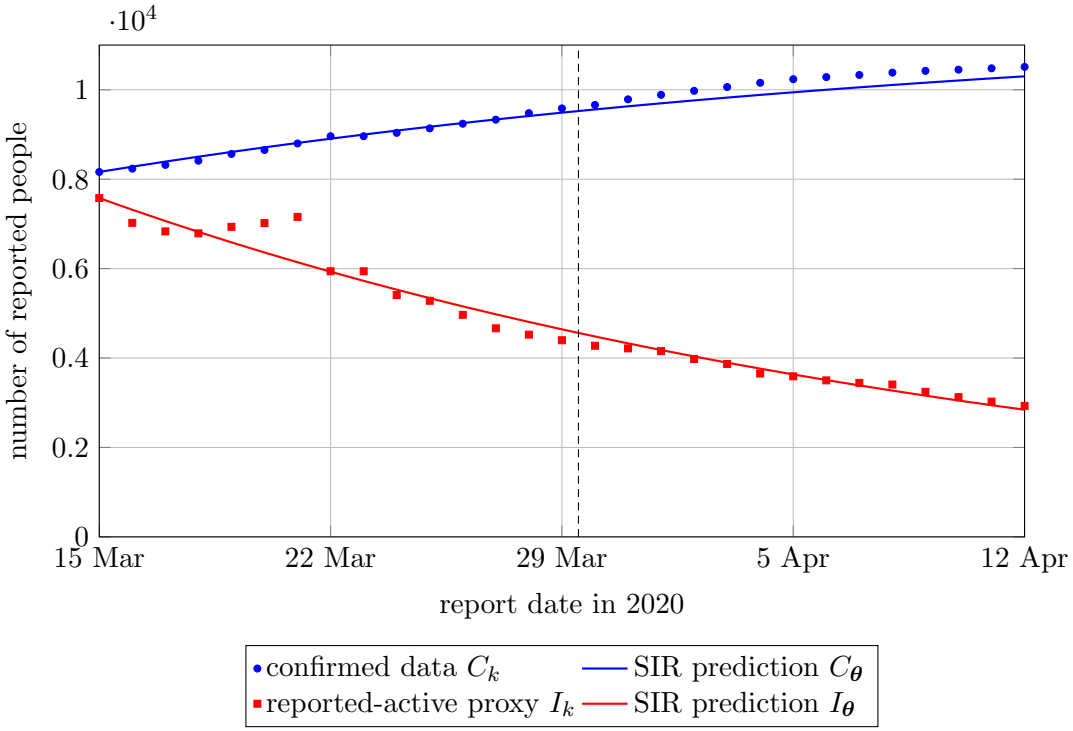


Figure 6.3.: South-Korean reported COVID-19 data and the SIR trajectory. Only observations to the left of the dashed line are used to estimate the parameters; points to its right form the untouched future control set.

This gives

$$\widehat{\beta} = 0.0158159 \text{ day}^{-1}, \qquad \widehat{\gamma} = 0.0508105 \text{ day}^{-1}. \tag{6.15}$$

The displayed digits reproduce the computation and the plotted curve; they are not a claim of seven-digit epidemiological precision. Thus $1/\widehat{\gamma} = 19.68$ days and $\widehat{\beta}/\widehat{\gamma} = 0.3113$. Since this window lies in the declining tail of the first wave and S/N is almost one, the last number describes the local reporting-and-intervention regime. We therefore cannot interpret it as the uncontrolled biological $\mathcal{R}_0$ of SARS-CoV-2.

6.7.3. Verification on future data

After obtaining (6.15), we freeze the parameters, the observation map, the initial condition, and all preprocessing choices. We then make a fresh RK45 integration from the original 15 March state over $[0, 28]$, using the same tolerances and all the same settings. It requires 31 accepted steps and 188 right-hand-side evaluations. Thus we do not restart the solution from the observed state on 29 March. For n control observations, we report

$$\text{RMSE} = \left(\frac{1}{n} \sum_{k=1}^{n} (\widehat{y}_k - y_k)^2 \right)^{1/2}, \qquad \text{MAPE} = \frac{100}{n} \sum_{k=1}^{n} \frac{|\widehat{y}_k - y_k|}{|y_k|}.$$

quantity	12 Apr. data	12 Apr. SIR	SIR RMSE	SIR MAPE	persistence RMSE
confirmed C	10512	10300.6	237.4	2.26%	659.9
active proxy I	2930	2843.9	93.8	2.22%	904.3
removed R	7582	7456.6	248.3	3.70%	1561.5

Table 6.3.: Forecast scores on the 14 daily observations that were not used in the inverse problem. All RMSE values are numbers of reported people.

The results are shown below. The persistence baseline simply holds the 29 March observation constant throughout the future window.

The model improves on this simple baseline for all three reported series, and the future curve in Figure 6.3 continues in the correct direction. This is meaningful evidence that the two estimated coefficients summarize this short declining regime. It is not evidence that a constant-parameter homogeneous SIR model explains the entire South-Korean epidemic.

In particular, recovery reports in this period are delivered in batches; M_k jumps by 627 on 16 March and by 1378 on 22 March. The active series $C_k - M_k$ is therefore not a direct measurement of latent infectious prevalence. Confirmed cases depend on testing and case definitions, infections are under-ascertained, imported cases violate closure, and behavior and isolation make β time dependent. The historical dashboard itself documents the difficulty of standardizing recovery and active-case data [6]. These are errors in the observation and modeling layers, not errors that a more accurate ODE solver can remove.

6.7.4. Extending this idea to a more general workflow

We can use the same procedure for other time-series problems, including applications unrelated to epidemics. The main steps are the following:

1. choose the model, observation operator, preprocessing, score, and chronological cutoff before fitting;
2. calibrate using only the prefix of the time series and retain parameter uncertainty, not only a point estimate;
3. freeze every modeling choice and integrate through the unseen suffix without resetting the state or refitting;
4. compare several observables and a simple baseline, and inspect residuals for trends, delay, and structural breaks;
5. report the control result, then—and only then—refit all data if a final descriptive estimate is desired.

For forecasting from time series, a random train–test split is inappropriate: it allows future observations to influence a fit presented as a forecast. Centered smoothing across

the cutoff causes the same leakage more subtly. A random split may instead be appropriate for data without a natural ordering, as in image recognition.

Finally, we complement data validation with numerical verification. We check nonnegativity and conservation of total population along the computed trajectory. Runge–Kutta methods preserve the linear population invariant in exact arithmetic, so its numerical variation should be attributable to accumulated roundoff. The nonlinear first integral (6.4) is generally not preserved exactly. We monitor its drift and verify that it decreases as we refine the integration tolerances, provided roundoff does not dominate.

We also verify that tighter ODE tolerances do not materially change the estimated parameters or fitted observables. Repeating the optimization from several initial guesses provides evidence about sensitivity to the starting point. Agreement supports the reproducibility of the solution but does not prove global optimality. Different outcomes can correspond to distinct local minima or poorly identifiable parameters, as well as to incomplete convergence. We therefore compare final objective values, termination diagnostics, and fitted trajectories before drawing a conclusion. These checks help us distinguish numerical error from limitations of the model and of the available observations.

9. Road networks and graph models

In this chapter we consider the problem of describing the structure of a road network and identifying its most important intersections and streets. The meaning of importance depends on the question under consideration. For instance, an intersection may offer many immediate choices, be easy to reach from the rest of a city, or belong to routes connecting distant neighbourhoods. Similarly, closing a road may cause a long detour even when it belongs to few shortest routes. We shall use graph models to formulate these questions. The same mathematical tools also apply to communication networks, interactions between individuals, and scientific collaborations.

As in Chapter 6, we first study the mathematical models independently of a particular dataset, and then consider their application to a numerical experiment. Section 9.1 develops graph representations and their matrix structure. Section 9.2 then develops a general framework for centrality and its principal instances. The definitions specify what a score measures and the assumptions under which it is meaningful. Section 9.3 studies diffusion and its numerical approximation. Section 9.4 assesses network interventions through changes in performance. Section 9.5 applies these ideas to Pisa and guides the accompanying notebooks. We remark that a structural score does not, by itself, predict traffic. Such a prediction would also require models for demand, capacity, congestion, and driver behaviour.

9.1. Graphs: mathematical foundations

9.1.1. Graphs, weights, and representations

We start by recalling the graph representations that will be needed in the following sections.

Definition 9.1.1.1. A finite simple undirected graph is a pair $G = (V, E)$, where $V = \{1, \dots, n\}$ and E is a set of unordered pairs of distinct vertices. We write $m = |E|$ and assume $n \geq 2$. A directed graph instead has ordered edges (i, j); a multigraph allows distinct edges with the same endpoints. Self-loops are edges from a vertex to itself.

Unless explicitly stated otherwise, the undirected theory below concerns simple graphs without self-loops. This convention avoids ambiguities in degrees and diagonal entries of matrices.

Before constructing a graph, we need to specify what its vertices and edges represent. This choice is part of the model. In a *primal* road graph, vertices are intersections or dead ends and edges are road segments. Legal travel directions give a directed graph, and

distinct links between the same intersections give parallel edges. Alternatively, streets can be vertices, with adjacency meaning that two streets intersect. This gives the so-called *dual street graph*. Its paths describe sequences of streets, rather than physical travel distances, and therefore answer a different modelling question. More generally, one can form an intersection graph from a family of objects by declaring two objects adjacent when they intersect. The definition of an object, such as a street, must be specified before applying any centrality measure.

Given a simple graph, we define its *adjacency matrix* as follows:

$$A_{ij} = \begin{cases} 1, & \{i,j\} \in E, \\ 0, & \text{otherwise.} \end{cases}$$

Thus $A = A^T$, $A_{ii} = 0$, and the degree is $d_i = \sum_j A_{ij}$. For a directed graph we use the convention $A_{ij} = 1$ for an edge $i \to j$: row sums are out-degrees and column sums are in-degrees. For a multigraph, integer entries can count parallel edges.

We will need to distinguish two meanings of an edge weight. A *cost* $\ell_e > 0$ is added along a route, so large values discourage using an edge. An *interaction weight* $w_e > 0$ describes the coupling between two vertices; a conductance is one such weight. Let $W_{ij} = w_{\{i,j\}}$ on an undirected edge and zero otherwise, and let $s_i = \sum_j W_{ij}$ be the *weighted degree* of vertex i. Replacing A by W in an adjacency-based centrality treats a large entry as a strong connection. Consequently, using a distance or travel time as an entry of W would require a separate modelling choice. The unweighted case is recovered with $W = A$.

9.1.2. Walks, paths, and distances

Definition 9.1.2.1. A walk of length k is a sequence $v_0, \dots, v_k$ with an edge from v_{r-1} to v_r at every step. Vertices and edges may repeat; a walk is closed when $v_0 = v_k$. A path has no repeated vertices. In a simple undirected graph, a cycle is a closed walk of length at least three whose vertices are distinct except for its identical first and last vertices. In a directed graph all steps must follow the edge directions. An undirected graph is connected when every pair of vertices is joined by a path. A directed graph is strongly connected when this holds for every ordered pair, and weakly connected when its underlying undirected graph is connected.

The adjacency matrix provides a convenient way to count walks, as shown by the following result.

Lemma 9.1.2.2. *For every integer $k \geq 0$, $(A^k)_{ij}$ counts walks of length k from i to j. In particular, $(A^k)_{ii}$ counts closed walks based at i.*

Proof. For $A^0 = I$, we count the unique length-zero walk at each vertex. If the assertion holds at k, then

$$(A^{k+1})_{ij} = \sum_{h=1}^{n} (A^k)_{ih} A_{hj}.$$

The term indexed by h counts all length-k walks from i to h followed by the edge $h \to j$. Every length-$(k+1)$ walk has exactly one such penultimate vertex, which proves the assertion by induction. □

For a multigraph the same argument counts choices of parallel edges; for W it gives a sum of products of edge weights along walks, rather than an integer count.

For a simple undirected graph, $(A^2)_{ii} = d_i$, whereas $(A^3)_{ii}$ is twice the number of triangles containing i. Consequently,

$$\operatorname{tr}(A^2) = 2m, \qquad \operatorname{tr}(A^3) = 6\,\#\{\text{triangles in } G\}.$$

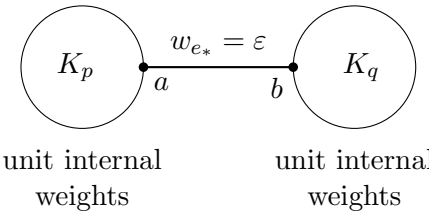
Remark 9.1.2.3. A closed walk need not be a cycle. For instance, traversing one edge and immediately returning gives a closed walk of length two. We will need this distinction when interpreting centralities defined through matrix functions.

If the edges have positive costs, we can define the distance between two vertices by

$$d_\ell(i,j) = \min_{P : i \rightsquigarrow j} \sum_{e \in P} \ell_e, \qquad d_\ell(i,i) = 0,$$

and set $d_\ell(i,j) = \infty$ if no path exists. Positivity ensures that a shortest walk between distinct vertices can be chosen to be a path, since removing a repeated-vertex segment strictly decreases its cost. On a connected undirected graph d_ℓ is a metric; directed distances need not be symmetric. Unit costs give hop distance. Breadth-first search computes unit-cost distances from one source in $O(n+m)$ operations; for positive costs, Dijkstra's algorithm with a binary heap uses $O((n+m)\log n)$ operations. Sections 5.3–5.5 explain these searches and their data structures. We will use them when computing centralities based on distances.

Example 9.1.2.4 (Two clusters joined by a bridge). Let G_ε consist of complete graphs on disjoint vertex sets U and Z, of sizes $p, q \geq 3$, joined by one edge $e_* = \{a, b\}$ with $a \in U$ and $b \in Z$. We assign unit route cost to every edge, unit weight to edges within each cluster, and weight $\varepsilon > 0$ to the joining edge. In this example, distances and interaction weights correspond to independent modelling choices. We will use this graph to compare local importance, passage between regions, diffusion, and the effects of a closure.



The circles stand for complete subgraphs; all route costs, including that of the bridge, remain equal to one.

9.1.3. Bipartite graphs

Definition 9.1.3.1. An undirected graph $G = (V, E)$ is *bipartite* if its vertex set admits a partition $V = U \dot\cup Z$ such that every edge joins a vertex of U to a vertex of Z. There are no edges within either part.

Lemma 9.1.3.2. *A graph is bipartite if and only if it contains no cycle of odd length.*

Proof. If the graph is bipartite, each step switches parts, so every closed walk in a bipartite graph has even length. Conversely, we choose a root in each connected component and partition its vertices according to the parity of their hop distance from the root. An edge joining vertices of the same parity would produce an odd closed walk, using their paths to the root, and hence an odd cycle. Thus the partition is valid when no odd cycle exists. □

In practice, breadth-first search constructs the partition, or detects a conflict, in $O(n+m)$ time. Trees and even cycles are bipartite; odd cycles are not. In particular, the recurring graph G_ε is not bipartite, since each of its clusters contains a triangle.

Bipartite graphs naturally model relations between two types of objects: students and courses, authors and papers, or stops and transport lines. They retain information that can be lost when replacing a shared association by a direct link between objects of the same type. For example, a stop–line graph distinguishes two stops served by one common line from two stops served by several common lines. A road graph whose vertices are intersections need not be bipartite, since its topology may contain odd cycles. We therefore need to check this property when studying a road network; it is not part of the modelling assumptions.

If we order the vertices so that those in U come first, the adjacency matrix and its square take the form

$$A = \begin{pmatrix} 0 & R \\ R^T & 0 \end{pmatrix}, \qquad A^2 = \begin{pmatrix} RR^T & 0 \\ 0 & R^TR \end{pmatrix}.$$

Here $R_{ij} = 1$ when vertex $i \in U$ is adjacent to vertex $j \in Z$. Off-diagonal entries of RR^T count shared neighbours of vertices in U; its diagonal records their degrees. This suggests projecting the graph onto one type of object. We should keep in mind, however, that the resulting links represent two-step walks in the original graph. One object in Z adjacent to many vertices in U generates a clique in the projection, potentially changing the interpretation of subsequent centrality scores.

We can also relate the bipartite structure to the spectrum of the adjacency matrix. Setting $J = \operatorname{diag}(I_{|U|}, -I_{|Z|})$, we have $JAJ = -A$. Consequently, $Ax = \mu x$ implies $A(Jx) = -\mu(Jx)$: the adjacency spectrum is symmetric about zero. For a connected bipartite graph with at least one edge, both $\rho(A)$ and $-\rho(A)$ are eigenvalues. A random walk alternates between the two parts and cannot return after an odd number of steps. These properties explain the possible oscillations of power iteration and the periodicity of random walks encountered below when constructing centrality measures.

9.1.4. The graph Laplacian: connectivity, cuts, and spectrum

We now consider an undirected graph G with positive edge weights and define

$$D_w = \operatorname{diag}(s_1, \dots, s_n), \qquad L_w = D_w - W.$$

For unit edge weights this is the combinatorial Laplacian $L = D - A$. To express the Laplacian in terms of edge differences, we choose an arbitrary orientation for each undirected edge and let $B \in \mathbb{R}^{m \times n}$ have a $+1$ at the initial vertex and a -1 at the final vertex of each edge. We set $C = \operatorname{diag}(w_e)$, so B maps vertex values to edge differences and C multiplies each edge difference by its weight.

We can use B to build a factorization for the Laplacian. If an edge e is oriented from u to v, its row in B is $(\boldsymbol{e}_u - \boldsymbol{e}_v)^T$, where $\boldsymbol{e}_i$ denotes the ith coordinate vector of $\mathbb{R}^n$. Since C is diagonal, matrix multiplication gives

$$B^T C B = \sum_{e=(u,v)} w_e (\boldsymbol{e}_u - \boldsymbol{e}_v)(\boldsymbol{e}_u - \boldsymbol{e}_v)^T,$$

where the sum contains each undirected edge once, with its chosen orientation. The summand associated with e has only four nonzero entries: $+w_e$ in positions (u,u) and (v,v), and $-w_e$ in positions (u,v) and (v,u). Thus each edge adds its weight to the two endpoint diagonals and subtracts it in the two corresponding off-diagonal positions. Summing the contributions, we obtain

$$(B^T C B)_{ij} = \sum_{e \in E} w_e B_{ei} B_{ej} = \begin{cases} \displaystyle\sum_{e:\, i \in e} w_e = s_i, & i = j, \\ -W_{ij}, & i \neq j. \end{cases}$$

For $i \neq j$, only an edge joining i and j can contribute, and its two incidence entries have opposite signs; if there is no such edge, the entry is zero. These are precisely the entries of $D_w - W$, proving

$$L_w = B^T C B.$$

The same representation also gives the quadratic form associated with the Rayleigh quotient directly. For an edge oriented from u to v, $(Bx)_e = x_u - x_v$. Consequently,

$$x^T L_w x = (Bx)^T C (Bx) = \sum_{\{i,j\} \in E} w_{\{i,j\}} (x_i - x_j)^2.$$

Reversing an edge orientation changes the sign of its row in B, but leaves both its outer-product contribution and the squared difference unchanged. Hence the factorization is independent of these arbitrary orientations. Since C has positive diagonal entries, L_w is symmetric positive semidefinite. Moreover, $B\mathbf{1} = 0$, so $L_w \mathbf{1} = B^T C B \mathbf{1} = 0$. We have thus recovered the same structure based on differences across edges as in the heat-diffusion operator of Chapter 8.

The nullspace of the Laplacian has a direct interpretation in terms of connected components.

Lemma 9.1.4.1. *If G has q connected components $V_1, \dots, V_q$, then*

$$\ker L_w = \operatorname{span}\{\mathbf{1}_{V_1}, \dots, \mathbf{1}_{V_q}\}, \qquad \dim \ker L_w = q.$$

Here $\mathbf{1}_S$ is the indicator vector of a vertex set S.

Proof. If $L_w x = 0$, then $x^T L_w x = 0$. All weights are positive, so the quadratic form forces $x_i = x_j$ along every edge and hence along every path. Thus x is constant on each component. Conversely, any such vector satisfies $Bx = 0$ and therefore $L_w x = 0$. The component indicators are linearly independent, proving the claim. Since L_w is symmetric, the multiplicity of its zero eigenvalue is exactly the number of components. □

We write its eigenvalues as $0 = \lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$. In particular, G is connected if and only if $\lambda_2 > 0$. The Courant–Fischer characterization gives

$$\lambda_2 = \min_{\substack{x \neq 0 \\ x^T \mathbf{1} = 0}} \frac{x^T L_w x}{x^T x}.$$

We shall now use the spectrum to study the problem of dividing the graph into two parts. In the following discussion we assume that G is connected. A bipartition is a pair $(S, V \setminus S)$ with $\varnothing \neq S \neq V$. We define its cut weight, counting each crossing edge once, as

$$w(\partial S) = \sum_{i \in S} \sum_{j \in V \setminus S} W_{ij}.$$

Minimizing this weight alone can favour separating a single vertex. To penalize small parts, consider the two-way *ratio-cut* objective

$$\mathrm{RC}(S) = w(\partial S) \left(\frac{1}{|S|} + \frac{1}{n - |S|} \right), \qquad \mathrm{RC}_* = \min_{\varnothing \neq S \neq V} \mathrm{RC}(S). \tag{9.1}$$

For a fixed cut weight, the factor in parentheses is smallest for parts as equal in size as possible. Note that we are penalizing imbalance, without imposing equal cardinalities. A minimum bisection with a prescribed part size is a different optimization problem.

In order to relate this discrete problem to an eigenvalue problem, we associate a vector with each partition. For each nonempty proper S, put $a = |S|$, $b = n - a$, and define

$$(z_S)_i = \begin{cases} \sqrt{b/(na)}, & i \in S, \\ -\sqrt{a/(nb)}, & i \notin S. \end{cases}$$

A direct computation gives $z_S^T \mathbf{1} = 0$ and $\|z_S\|_2 = 1$. The difference across every crossing edge is $\sqrt{n/(ab)}$, while all differences within either part vanish. Hence

$$z_S^T L_w z_S = \frac{n}{ab} w(\partial S) = \mathrm{RC}(S).$$

If $\mathcal{Z} = \{z_S : \varnothing \neq S \neq V\}$, the partition problem therefore has the exact formulation

$$\mathrm{RC}_* = \min_{z \in \mathcal{Z}} z^T L_w z.$$

We obtain a spectral relaxation by dropping the requirement that z have the prescribed two-level form, while retaining its norm and orthogonality constraints:

$$\min_{\substack{x^T \mathbf{1} = 0 \\ \|x\|_2 = 1}} x^T L_w x = \lambda_2 \leq \mathrm{RC}_*. \tag{9.2}$$

To prove the equality and characterize its minimizers, we choose an orthonormal Laplacian eigenbasis with $q_1 = \mathbf{1}/\sqrt{n}$. Every feasible vector has the expansion $x = \sum_{r=2}^{n} \alpha_r q_r$, where $\sum_{r=2}^{n} \alpha_r^2 = 1$. Thus

$$x^T L_w x = \sum_{r=2}^{n} \lambda_r \alpha_r^2 \geq \lambda_2,$$

with equality exactly when x lies in the eigenspace of λ_2. A unit vector in this eigenspace is called a *Fiedler vector*. The inequality in (9.2) follows from $\mathcal{Z} \subset \{x : x^T \mathbf{1} = 0,\ \|x\|_2 = 1\}$. In particular, for every cut,

$$\lambda_2 \leq \mathrm{RC}(S) = \frac{n\, w(\partial S)}{|S|(n - |S|)}. \tag{9.3}$$

This shows why a weak cut between substantial parts implies a small λ_2. We remark that the relaxed and discrete problems have different feasible sets. A minimizer of the former need not solve the latter. This is the unnormalized spectral relaxation of ratio cut; see [24] for its relation to other spectral clustering constructions.

Having computed a Fiedler vector, we still need to recover a discrete partition. A possible strategy is to consider all its threshold cuts. Let v be a unit Fiedler vector, and order the vertices by a permutation π such that $v_{\pi(1)} \leq \cdots \leq v_{\pi(n)}$. Define the prefixes

$$S_k = \{\pi(1), \ldots, \pi(k)\}, \quad k = 1, \ldots, n, \qquad S_0 = \varnothing.$$

We retain as candidate cuts only the indices in

$$\mathcal{K} = \{k \in \{1, \ldots, n-1\} : v_{\pi(k)} < v_{\pi(k+1)}\}.$$

The sets S_k with $k \in \mathcal{K}$ are exactly the distinct nonempty proper sublevel sets $\{i : v_i \leq t\}$ as t varies. Equal entries are kept together, so arbitrary ordering within a tied block does not create extra candidates. Because $v \neq 0$ and $v^T \mathbf{1} = 0$, it has both positive and negative entries, and at least one such cut exists. We then select

$$k_* \in \arg\min_{k \in \mathcal{K}} \mathrm{RC}(S_k), \qquad \widehat{S} = S_{k_*}.$$

The resulting partition minimizes the discrete objective among the threshold cuts of the chosen vector. This is a smaller set than the collection of all bipartitions. Cutting at

zero is only one candidate and need not minimize this sweep objective. After sorting, all cut weights can be updated by adding one vertex at a time:

$$c_{k+1} = c_k + s_{\pi(k+1)} - 2 \sum_{j \in S_k} W_{\pi(k+1),j}, \qquad c_0 = 0,$$

for $k = 0, \ldots, n-1$, where $c_k = w(\partial S_k)$; evaluate candidates only after completing a tied block. By scanning the adjacency lists, we obtain a total sweep cost $O(n \log n + m)$, excluding the eigenvector computation.

Remark 9.1.4.2. We can assess the quality of the partition by comparing its objective value with the relaxed minimum. If the exact eigenvalue is available, we have

$$\lambda_2 \le \mathrm{RC}_* \le \mathrm{RC}(\widehat{S}), \qquad 0 \le \mathrm{RC}(\widehat{S}) - \mathrm{RC}_* \le \mathrm{RC}(\widehat{S}) - \lambda_2.$$

Consequently, a small gap guarantees that the objective value is close to the minimum; equality implies that the partition is optimal. In particular, if a Fiedler vector has exactly two values, orthogonality and unit norm make it z_S for its positive level set S, up to sign. That partition attains $\mathrm{RC}_* = \lambda_2$. In general, the relaxation can have a nonzero gap and the sweep does not guarantee a globally optimal cut. If λ_2 is simple, the unit vector is unique up to sign; reversing the sign gives the same family of unordered candidate bipartitions. A repeated λ_2 allows different vectors and different sweep partitions. Even a unique vector can yield several minimizing thresholds. It is therefore necessary to specify the selected cut and its objective value, since a uniquely determined "Fiedler partition" need not exist."

In numerical computations, we need to check the eigenpair residual, orthogonality to $\mathbf{1}$, and separation from neighbouring eigenvalues. A computed Rayleigh quotient is an upper bound for the relaxed minimum, not a certified lower bound for the discrete optimum. A rigorous numerical optimality certificate therefore needs a validated lower bound on λ_2, rather than substituting an unchecked approximate eigenvalue in the preceding inequalities. Rescaling all edge weights by a positive constant rescales both λ_2 and every ratio-cut value, while leaving the minimizing partitions unchanged.

The same eigenvalues also describe diffusion on the graph. For a connected graph and $\dot{u} = -L_w u$, orthonormal eigenvectors q_r give

$$u(t) = \bar{u}\mathbf{1} + \sum_{r=2}^{n} e^{-\lambda_r t} q_r q_r^T u(0), \qquad \bar{u} = \frac{\mathbf{1}^T u(0)}{n}.$$

Thus the mean is preserved, and $\|u(t) - \bar{u}\mathbf{1}\|_2 \le e^{-\lambda_2 t}\|u(0) - \bar{u}\mathbf{1}\|_2$. On a disconnected graph there is a separate limiting mean on each component. The nullspace identifies components, while the first positive eigenvalues measure slow equilibration between regions. Section 9.3 develops the balance law, its qualitative properties, and numerical methods for simulating this process.

9.2. Centrality measures

9.2.1. Centrality as a modelling construction

In the preceding section we have introduced the graph representations and their main properties. We now consider how to assign an importance score to a vertex or an edge. This requires choosing what we mean by participation, accessibility, or influence. A vertex with many neighbours, a vertex close to many destinations, and a vertex carrying interactions between other pairs can therefore be central for different reasons. We need to specify the interaction model before interpreting the resulting ranking.

Definition 9.2.1.1. A node centrality is a rule assigning to each admissible graph G a vector $C_G \in \mathbb{R}^V$; an edge centrality assigns a vector in $\mathbb{R}^E$. The admissible class specifies directions, weights, and any connectivity assumptions. Parameters and external data, such as travel demand or a preferred distribution of starting vertices, are also inputs to the rule. Larger scores will mean greater importance for the chosen question. We require a structural rule to respect relabelling: if $\pi : V \to V'$ is a graph isomorphism preserving the weights and all other inputs, then

$$C_{\pi(G)}(\pi(i)) = C_G(i).$$

The analogous requirement holds for edges.

In particular, vertices exchanged by a symmetry preserving all inputs must receive equal scores. This condition ensures that the ranking reflects the model rather than the vertex labels; it does not single out a unique centrality.

We shall consider two ways of constructing centrality measures. The first consists in specifying a family Ω of interactions, a nonnegative weight a_ω for each interaction, and a nonnegative response $h_i(\omega)$ measuring how vertex i participates. We then define

$$S_G(i) = \sum_{\omega \in \Omega} a_\omega h_i(\omega), \qquad C_G(i) = \mathcal{N}_G(S_G)(i), \tag{9.4}$$

whenever the sum is finite or convergent. Here $\mathcal{N}_G$ is a specified transformation, such as division by a fixed population size, or taking the reciprocal after converting a sum of distances to a mean. This transformation must also respect relabelling. We remark that the formula is only informative once the interactions, responses, and transformation have been specified; allowing arbitrary choices would not define a particular model. For an edge score, we replace h_i by h_e. The interactions may be destinations, origin–destination pairs, walks, or realizations of a transport process. Thus choosing a centrality amounts to choosing what is exchanged or explored, how it moves, how participation is measured, and how contributions are aggregated. Equation (9.4) organizes this family of constructions; it is not a universal definition of centrality.

The second construction consists in describing how scores reinforce one another, and then solving the corresponding equilibrium equation. Examples are an eigenvector equation, a fixed point with a baseline input, and a pair of coupled hub–authority equations.

In this case, we need to establish existence and uniqueness for the equilibrium equation and specify a normalization. These two constructions can meet: a convergent series can express a fixed point as a sum over walks, and a spectral limit can select a dominant recursive score. Such connections must be derived for each model.

The centralities considered below can be related to these constructions as follows.

- *Local connections and accessibility.* Taking destinations $j \neq i$ and responses A_{ij} gives degree. Taking responses $d_\ell(i,j)$ and applying the reciprocal-mean transformation gives closeness on a connected graph. Summing responses $1/d_\ell(i,j)$ gives harmonic centrality.

- *Participation in routes.* We assign a demand weight to each reachable pair (s,t) and choose a shortest path uniformly among all tied shortest paths. The expected indicator that a vertex or edge lies on that route gives its betweenness contribution. Electrical models replace this route indicator by the magnitude of current carried for the same unit injection.

- *Participation in walks.* We take walks as interactions and weight them according to length. Counting walks starting at i gives Katz or total communicability scores for different attenuation rules; counting closed walks based at i gives subgraph centrality.

- *Recursive influence and stochastic occupation.* We define scores through an equilibrium equation: neighbours pass on importance, or a random walker redistributes probability. Eigenvector centrality, hub and authority scores, and PageRank arise this way. Spectral limits and convergent matrix series relate these equilibrium constructions to walk aggregation.

The aggregation and the final transformation have different meanings. For example, closeness takes the reciprocal *after* averaging distances, whereas harmonic centrality averages reciprocal distances. These operations can produce different rankings. Multiplying every score by the same positive constant preserves the ranking, but changes its units or normalization. Scores on different graph sizes, components, or weight scales need not be comparable even when each has been normalized. When the graph is disconnected or directed, we also need to specify how unreachable pairs are treated and in which direction interactions occur. Finally, centrality describes the chosen interaction model on the existing graph; the effect of removing an edge or vertex requires a separate performance comparison.

Exercise 9.2.1.2. Compare a path graph, a cycle, and a star on the same number of vertices. Which vertices should be important if the question concerns local choices, average distance, or the consequences of deleting a vertex?

9.2.2. Local centrality and accessibility

We start with two elementary notions of importance: the number of immediate connections and the ease of reaching the other vertices. These lead to different scores.

Definition 9.2.2.1. On a simple undirected graph with $n \geq 2$, the degree centrality is

$$C_D(i) = \frac{d_i}{n-1}.$$

This score measures immediate connections and lies in $[0, 1]$ for a simple undirected graph. The weighted degree s_i measures the total incident edge weight, without a universal normalization by $n - 1$. Directed graphs distinguish incoming and outgoing degrees, and likewise incoming and outgoing weighted degrees. Multigraph degrees count multiplicities, and the same normalization need not produce values below one.

Definition 9.2.2.2. On a connected undirected graph, closeness centrality is the reciprocal of the mean distance to other vertices:

$$C_C(i) = \frac{n-1}{\sum_{j \neq i} d_\ell(i, j)}.$$

In this case, we measure accessibility rather than local branching. On disconnected graphs the denominator can be infinite. Restricting it to a component is possible, but compares different populations; multiplying by the fraction of reachable vertices is another convention and must be stated explicitly. We can account for unreachable pairs directly by using harmonic centrality:

$$C_H(i) = \sum_{j \neq i} \frac{1}{d_\ell(i, j)}, \qquad \frac{1}{\infty} := 0.$$

We may normalize this score by $n - 1$. For directed graphs, the formulas with $d_\ell(i, j)$ measure outward accessibility, whereas using $d_\ell(j, i)$ measures accessibility *to* i. We need to specify which of these two conventions is being used. Both closeness and harmonic centrality have inverse-cost units; changing all lengths from metres to kilometres rescales scores but does not change rankings.

Example 9.2.2.3. In G_ε, the bridge endpoint b has degree q, whereas an ordinary vertex of U has degree $p - 1$. If $p > q + 1$, degree ranks the latter higher even though it does not connect the two regions. Unit route costs also mean that changing ε changes neither degree nor distance scores.

From a computational point of view, computing all degrees from adjacency lists costs $O(n + m)$. Repeated single-source searches compute accessibility scores without storing a dense all-pairs distance matrix.

9.2.3. Shortest-path betweenness

We now consider a different question: identifying the vertices and edges that mediate travel between other pairs, assuming that trips follow shortest routes. We fix positive edge costs and split each pair's unit demand equally among tied shortest paths. Let σ_{st}

denote the number of shortest paths from s to t and $\sigma_{st}(v)$ the number containing v as an internal vertex. In an undirected graph, the unnormalized node betweenness is

$$C_B(v) = \sum_{\substack{s<t \\ s,t\neq v}} \frac{\sigma_{st}(v)}{\sigma_{st}},$$

where an unreachable pair contributes zero. For an edge e, we define

$$C_B(e) = \sum_{s<t} \frac{\sigma_{st}(e)}{\sigma_{st}}.$$

These formulas assign one unit of interaction to every connected pair and split it equally between tied shortest paths. For $n > 2$, dividing node scores by $\binom{n-1}{2}$ is a common normalization; edge scores can be divided by $\binom{n}{2}$. Directed versions sum over ordered pairs, with normalizations $(n-1)(n-2)$ and $n(n-1)$. Nonuniform demand can be represented by multiplying each summand by a prescribed pair weight.

A bridge is an edge whose removal increases the number of components. If a bridge of a connected unweighted graph separates sets of sizes a and b, its edge betweenness is ab: every cross-set pair must use it, and no within-set shortest path does. For a non-bridge, a high score need not mean that removal is damaging; alternative routes may be only slightly longer.

Example 9.2.3.1. For G_ε, the bridge has edge betweenness pq, while its endpoints have node betweenness $(p-1)q$ and $p(q-1)$; every other vertex has zero node betweenness. Thus passage between clusters can give a vertex high betweenness even when its degree is modest.

The definition might suggest that we need to enumerate all shortest paths. Their number can be large, but this enumeration is not necessary. For a fixed source s, we store distances, counts σ_{sv}, and predecessors in the shortest-path directed acyclic graph. With positive costs, every predecessor has smaller distance. We then process vertices in reverse distance order using the dependencies

$$\delta_s(v) = \sum_{w:\, v \text{ is a predecessor of } w} \frac{\sigma_{sv}}{\sigma_{sw}}\big(1 + \delta_s(w)\big).$$

The fraction is the share of shortest paths to w that arrive through v; the term 1 accounts for destination w, and $\delta_s(w)$ accounts for destinations beyond it. Summing dependencies over sources, excluding the source itself, gives node scores, with a factor $1/2$ for undirected graphs. This is Brandes' algorithm [3]. Its unweighted cost is $O(nm)$ on a connected graph; a binary-heap weighted implementation costs $O(n(m+n)\log n)$. We may also estimate the scores by sampling sources and applying an appropriate rescaling. In this case, we need to check the stability of the ranking across samples.

9.2.4. Eigenvector and Katz centrality

Another possibility is to require the score of a vertex to be proportional to the sum of the scores of its neighbours. This gives the eigenvalue problem

$$Ax = \rho(A)x, \qquad x > 0, \qquad \|x\|_2 = 1.$$

For a connected undirected graph, Perron–Frobenius theory guarantees that the spectral radius $\rho(A) > 0$ is a simple eigenvalue with a strictly positive eigenvector. With the prescribed normalization, we obtain a unique eigenvector centrality. On a disconnected graph, components with smaller spectral radius receive zero weight in a leading eigenvector; ties between dominant components make it nonunique. In a directed graph, Ax propagates scores from outgoing neighbours, whereas $A^T x$ rewards incoming links. Strict positivity and uniqueness follow under strong connectivity.

We can modify this construction by introducing a baseline $\beta > 0$. The resulting score, known as Katz centrality, satisfies

$$x = \alpha Ax + \beta\mathbf{1}, \qquad x = \beta(I - \alpha A)^{-1}\mathbf{1}, \qquad 0 < \alpha < 1/\rho(A).$$

The Neumann series yields

$$x = \beta \sum_{k=0}^{\infty} \alpha^k A^k \mathbf{1}.$$

Thus the score counts all walks starting at a vertex, attenuating a length-k walk by α^k; for incoming walks use A^T. The baseline also gives positive scores on disconnected graphs. For small α, $x = \beta\mathbf{1} + \alpha\beta d + O(\alpha^2)$, so the first variation from a uniform score is degree. For a connected undirected graph, expansion in orthonormal adjacency eigenvectors q_r gives

$$x = \beta \sum_{r=1}^{n} \frac{q_r^T \mathbf{1}}{1 - \alpha\mu_r} q_r, \qquad Aq_r = \mu_r q_r, \quad \mu_1 = \rho(A).$$

As $\alpha \uparrow 1/\rho(A)$ the positive leading term dominates, so the normalized Katz vector converges to eigenvector centrality. At the same time the linear system becomes ill-conditioned.

Let us consider how to compute these scores. For a connected bipartite graph, $-\rho(A)$ is also an eigenvalue. Plain power iteration on A may then oscillate. A positive diagonal shift, such as $A + I$, retains the eigenvectors and removes this obstruction; alternatively, we may use a suitable eigensolver. A small residual $\|Ax - \rho x\|_2$ checks the computed eigenpair, while a small spectral gap can still make rankings sensitive to perturbations.

For Katz centrality, we first need an estimate of $\rho(A)$ to choose α. In practice, we solve the sparse linear system without forming an inverse. These constructions extend to nonnegative weight matrices W.

9.2.5. Matrix functions: communicability and closed walks

The series representation of Katz centrality suggests considering more general weights for walks of different lengths. Katz uses geometric attenuation; by introducing matrix functions, we can choose other weights and distinguish walks ending anywhere from closed walks. For a convergent power series $f(z) = \sum_{k\geq 0} a_k z^k$ with $a_k \geq 0$, the vector $f(A)\mathbf{1}$ counts attenuated walks from each vertex. Its diagonal instead measures closed walks. The exponential gives

$$SC(i) = (e^A)_{ii} = \sum_{k=0}^{\infty} \frac{(A^k)_{ii}}{k!}, \qquad TC(i) = (e^A\mathbf{1})_i.$$

The first is *subgraph centrality* [9]; the second is *total communicability*. Both series converge for every finite matrix. Subgraph centrality includes repeated traversals, not just simple cycles or distinct subgraphs. The identity term contributes one at every vertex; for a simple graph the next nonzero diagonal contribution is $d_i/2$.

For symmetric $A = Q\,\mathrm{diag}(\mu_r)Q^T$,

$$SC(i) = \sum_{r=1}^{n} e^{\mu_r} Q_{ir}^2.$$

Applying the power series to the diagonalization gives this spectral interpretation of the score. A parameter $\tau > 0$ in $e^{\tau A}$ controls the emphasis on long walks. For connected graphs, large τ makes the diagonal dominated by $e^{\tau\rho(A)}Q_{i1}^2$, whereas small τ emphasizes local structure. With weighted adjacency, the scale of W and the choice of τ must be considered together.

We can compute these scores by a dense symmetric diagonalization, at a cost of $O(n^3)$ operations and $O(n^2)$ storage. For large sparse graphs, Krylov approximation can compute $e^{\tau A}\mathbf{1}$ without forming the full exponential. This does not directly provide its diagonal: subgraph centrality instead needs methods for individual quadratic forms $e_i^T e^{\tau A} e_i$, such as Lanczos quadrature, or diagonal estimators. We therefore need different computational procedures for the two scores.

The exponential $e^{\tau A}$ amplifies and aggregates weighted walks; its row sums need not equal one. It differs from the diffusion operator e^{-tL_w} studied in Section 9.3, which redistributes values while preserving constants and, for symmetric weights with unit capacities, total amount. Although both constructions involve a matrix exponential, they correspond to different models.

9.2.6. Hubs, authorities, and HITS

For a directed graph, we may distinguish two roles for a vertex: selecting useful destinations and being selected by useful sources. In a graph of web pages, a good *hub* points to good *authorities*, while a good authority receives links from good hubs. This mutual reinforcement is the basis of Kleinberg's *Hyperlink-Induced Topic Search* (HITS) algorithm [16]. It assigns two scores to each vertex rather than a single recursive importance score.

We retain the convention $A_{ij} = 1$ for $i \to j$, and assume $A \neq 0$. Let $h, a \geq 0$ denote the hub and authority vectors, each normalized to unit Euclidean norm. The coupled equations are

$$\sigma h = Aa, \qquad \sigma a = A^T h.$$

Thus h_i aggregates the authority scores of outgoing neighbours, while a_j aggregates the hub scores of incoming neighbours. Substitution gives

$$AA^T h = \sigma^2 h, \qquad A^T A a = \sigma^2 a.$$

HITS chooses $\sigma = \sigma_1(A)$, the largest singular value of A; h and a are corresponding left and right singular vectors. Nonnegative choices exist because the two product matrices have nonnegative entries. Equivalently, the scores solve

$$\max_{\substack{\|h\|_2 = \|a\|_2 = 1 \\ h, a \geq 0}} h^T A a.$$

Indeed, Cauchy–Schwarz bounds $h^T A a$ by $\|Aa\|_2 \leq \sigma_1(A)$, and a nonnegative leading singular pair attains the bound. This is another instance of the equilibrium construction introduced above, now with two mutually dependent roles.

We can interpret the two product matrices by looking at their entries:

$$(A^T A)_{ij} = \sum_k A_{ki} A_{kj}, \qquad (AA^T)_{ij} = \sum_k A_{ik} A_{jk}.$$

For distinct vertices, the first counts common incoming neighbours and the second common outgoing neighbours. Their diagonal entries are in-degrees and out-degrees, respectively. Alternatively, we can construct two copies of every vertex, corresponding to its hub and authority roles. We join hub copy i to authority copy j whenever $i \to j$. This bipartite graph has adjacency matrix

$$\mathcal{A} = \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix}, \qquad \mathcal{A} \begin{pmatrix} h \\ a \end{pmatrix} = \sigma_1(A) \begin{pmatrix} h \\ a \end{pmatrix}.$$

The construction in Section 9.1.3 therefore connects HITS to eigenvector centrality on a graph of roles. Even when the original graph is connected, this bipartite graph need not be connected.

Example 9.2.6.1. Let us consider the graph with only the edges $1 \to 3$ and $2 \to 3$. We obtain $h = (1, 1, 0)^T/\sqrt{2}$, $a = (0, 0, 1)^T$, and $\sigma_1 = \sqrt{2}$: the two recommending vertices are hubs and their common destination is an authority. This also shows that HITS scores need not be strictly positive.

For weighted graphs, we replace A by a nonnegative weight matrix W; using travel costs requires a separate modelling decision. Unlike PageRank below, HITS does not divide a source's contribution among its outgoing neighbours or include teleportation. In the original search setting the graph is selected around a topic, so the selected subgraph is part of the definition of the ranking.

For road networks, hub and authority scores describe outgoing and incoming connectivity patterns, rather than measured trip production or attraction. They are most informative when direction or two distinct types of objects have a meaningful role. On an undirected graph $A^TA = AA^T = A^2$, so the two roles lose their directional distinction. If the graph is connected and non-bipartite, both scores reduce to its positive leading adjacency eigenvector. On a connected bipartite graph, the eigenvalues $\rho(A)$ and $-\rho(A)$ give the same squared eigenvalue, so the HITS leading singular value is repeated and the preceding uniqueness conclusion no longer holds.

We can compute the scores by an alternating iteration. Starting from $h^{(0)} > 0$, we repeatedly compute

$$a^{(k+1)} = \frac{A^T h^{(k)}}{\|A^T h^{(k)}\|_2}, \qquad h^{(k+1)} = \frac{Aa^{(k+1)}}{\|Aa^{(k+1)}\|_2}.$$

One full step is normalized power iteration on AA^T. Its eigenvalues are squared singular values and hence nonnegative. If $\sigma_1 > \sigma_2$, the positive initial vector has a nonzero projection onto the nonnegative leading eigenvector, and the iterates converge to the unique nonnegative leading singular pair. The asymptotic error factor is $(\sigma_2/\sigma_1)^2$. If the largest singular value is repeated, the result can depend on initialization; a small gap makes scores sensitive to perturbations. Strong connectivity of the original directed graph alone does not guarantee uniqueness: a directed cycle has $AA^T = I$. Given computed unit vectors, we can estimate $\sigma = h^TAa$ and check both $\|Aa - \sigma h\|_2$ and $\|A^Th - \sigma a\|_2$. Each iteration uses two sparse matrix–vector products, costing $O(n+m)$; forming the generally denser product matrices is unnecessary.

9.2.7. Random walks and PageRank

We now consider a random walker that chooses among its outgoing neighbours with probabilities proportional to their edge weights. We are interested in the vertices receiving a large long-run share of its visits. To this end, we assume nonnegative edge weights W on a directed graph and choose a distribution for restarting the walk. We form a row-stochastic transition matrix by $P_{ij} = W_{ij}/s_i^{\text{out}}$ when $s_i^{\text{out}} = \sum_j W_{ij} > 0$. At a vertex with no outgoing edge, we replace the row by a prescribed probability vector p^T. A column probability vector evolves by $z_{k+1} = P^Tz_k$.

Let us fix $0 < \gamma < 1$ and $p \geq 0$ with $\mathbf{1}^Tp = 1$. The PageRank vector satisfies

$$r = \gamma P^Tr + (1-\gamma)p, \qquad r = (1-\gamma)(I - \gamma P^T)^{-1}p.$$

Here $\rho(P) = 1$, so the inverse exists and its Neumann expansion shows $r \geq 0$. Multiplication by $\mathbf{1}^T$ shows $\mathbf{1}^Tr = 1$. If every entry of p is positive, the corresponding transition matrix $\gamma P + (1-\gamma)\mathbf{1}p^T$ is positive: the interpretation is a random walk with occasional teleportation to a new vertex sampled from p.

For a connected undirected weighted graph without teleportation, $\pi_i = s_i/\sum_j s_j$ is stationary, since $\pi_iP_{ij} = W_{ij}/\sum_j s_j = \pi_jP_{ji}$. This explains why PageRank and degree can be closely related in undirected networks. A stationary vector need not be the

limit of the ordinary walk: on a connected bipartite graph, probability can alternate between the two parts. Teleportation removes this obstruction. PageRank differs from Katz because outgoing scores are divided among neighbours through P; it also permits a nonuniform baseline encoding external information.

To compute the vector, we may use the fixed-point iteration. Its error satisfies

$$\|r_{k+1} - r\|_1 \le \gamma \|r_k - r\|_1,$$

since $\|P^T\|_1 = 1$. Thus the solution is unique and iteration converges regardless of connectivity or periodicity. Each iteration uses a sparse multiplication by P^T, together with the restart contribution. Dangling rows can be applied implicitly, so a step costs $O(n + m)$ without assembling a dense transition matrix.

9.2.8. Electrical flows and effective resistance

Shortest-path betweenness restricts interaction to minimum-cost routes. An alternative model allows it to spread over an undirected connected network. We assign conductances $c_e > 0$, set $C = \operatorname{diag}(c_e)$, and use the incidence convention of Section 9.1.4. For one unit injected at s and extracted at t, we set $b = e_s - e_t$. Ohm's and Kirchhoff's laws are

$$I = CB\phi, \qquad B^T I = b, \qquad L_c\phi = b, \quad L_c = B^T C B.$$

The nullspace result shows that this system is solvable, because $b^T\mathbf{1} = 0$, and that potentials are unique up to a constant. Grounding one vertex produces a positive definite reduced system. Equivalently, $\phi = L_c^+ b$ is the zero-mean solution, where L_c^+ is the Moore–Penrose pseudoinverse. Currents do not depend on the choice of gauge.

Lemma 9.2.8.1. *The electrical flow minimizes the dissipated energy among flows with the same injection.*

Proof. If J is any other flow with $B^T J = b$, we write $J = I + z$, so that $B^T z = 0$. Since $C^{-1} I = B\phi$,

$$J^T C^{-1} J = I^T C^{-1} I + z^T C^{-1} z \ge I^T C^{-1} I,$$

where the cross term vanishes by $\phi^T B^T z = 0$. □

This gives a variational interpretation of how the flow uses alternative routes.

Effective resistance is the voltage difference for unit injection:

$$R_{st} = b^T L_c^+ b = \phi_s - \phi_t = I^T C^{-1} I.$$

If $L_c q_r = \lambda_r q_r$ with orthonormal eigenvectors, then

$$R_{st} = \sum_{r=2}^{n} \frac{(q_r(s) - q_r(t))^2}{\lambda_r}.$$

Small positive Laplacian eigenvalues can therefore produce large resistance between regions. On a tree there is a unique route and R_{st} is the sum of edge resistances $1/c_e$ on

it. Adding an edge cannot increase effective resistance: the old flow with zero current on the added edge remains feasible in the energy minimization. This explains the role of redundant connections without assuming that all traffic follows an electrical law.

Example 9.2.8.2. In G_ε, unit current between a vertex in U and one in Z must cross e_*. Its contribution to the dissipated energy is $1/\varepsilon$, and $R_{ab} = 1/\varepsilon$. Weakening this conductance therefore increases resistance even though unit-cost shortest paths and their betweenness scores stay unchanged.

We write I_e^{st} for the current for pair s, t. Explicit unnormalized current-flow betweenness conventions are

$$C_{\mathrm{CF}}(e) = \sum_{s<t} |I_e^{st}|, \qquad C_{\mathrm{CF}}(v) = \sum_{\substack{s<t \\ s,t\neq v}} \frac{1}{2} \sum_{e \text{ incident to } v} |I_e^{st}|.$$

The factor $1/2$ avoids counting both arrival and departure at an internal vertex. Libraries may normalize these sums differently, so raw scores require matching conventions. This construction is related to random-walk betweenness [19]; it describes net electrical currents, not the total number of repeated random-walk traversals. Another resistance-based accessibility score is $(n-1)/\sum_{t\neq s} R_{st}$, often called current-flow closeness up to normalization.

For a disconnected graph, we need to work on each component and specify which pairs are included: a source and sink in different components cannot sustain the prescribed flow. Applying a pseudoinverse formula blindly to such a pair does not give its physical resistance.

For the computation, we can reuse a grounded Laplacian factorization for multiple right-hand sides. Iterative solvers and sampled pairs provide alternatives for larger graphs. In either case, we avoid forming a dense pseudoinverse explicitly.

9.2.9. Choosing a centrality measure

The choice of a centrality measure depends first on the question under consideration and on the meaning of the weights. We can then check its assumptions and computational cost. Table 9.1 summarizes the constructions above. Except where a probability distribution is intended, normalized values from different families need not have a common scale. We can instead compare rankings and interpret their differences in terms of the underlying models. For instance, on G_ε with $p = 6$, $q = 3$, the smaller cluster's bridge endpoint has degree 3 and node betweenness 12, while an ordinary vertex of the larger cluster has degree 5 and node betweenness 0. Neither ranking contradicts the other: they measure different roles.

9.3. Diffusion on graphs

So far we have assigned scores to a fixed network. We now turn to the study of a quantity that evolves through local exchanges along its edges. This leads us to diffusion

Table 9.1.: Centrality measures: modelling choices and numerical tasks.

Question and interaction model	Weights and main assumptions	Numerical task
Degree/weighted degree: immediate connections	Adjacency or edge weights; incoming/outgoing convention	Row/column sums
Closeness/harmonic: access to destinations	Positive costs; direction and unreachable-pair rule	Repeated shortest-path searches
Betweenness: participation in shortest routes	Positive costs; pair demand; split tied paths	Path counts and dependencies
Eigenvector: recursive reinforcement	Nonnegative edge weights; connectivity for uniqueness	Leading eigenvector
Katz: geometrically attenuated walks	Edge weights; $\alpha\rho(W) < 1$, positive baseline	Sparse linear solve
Communicability/subgraph: walks or closed walks	Edge weights; attenuation scale	Exponential action or diagonal estimation
HITS: coupled hub and authority roles	Two roles; simple leading singular value for uniqueness	Leading singular pair
PageRank: occupation with restarts	Transition and restart rules; $0 < \gamma < 1$	Contractive fixed-point iteration
Current-flow: passage or resistance accessibility	Undirected conductances; connected pairs; demand	Grounded Laplacian solves

on graphs. Examples include heat exchange between connected rooms, mixing between reservoirs, and agreement between agents who repeatedly adjust their values towards those of their neighbours. These interpretations lead to the same type of linear balance law. However, we need to specify the meaning and units of the state and edge coefficients for each application. On a road graph, diffusion can represent an idealized spreading process; vehicle motion with prescribed destinations, one-way restrictions, and congestion generally requires a different model.

9.3.1. From edge exchanges to a balance law

Let us consider an undirected graph with symmetric conductances $w_{ij} > 0$ on edges, and let $w_{ij} = 0$ otherwise. Let $u_i(t)$ be an intensive quantity, such as temperature or concentration, and let $m_i > 0$ be the storage capacity at vertex i. The stored amount is $m_i u_i$. We assume that the flux from i to j is $w_{ij}(u_i - u_j)$: exchange is proportional to the difference and directed from larger to smaller values. If we also prescribe a source $f_i(t)$, conservation at each vertex yields

$$m_i \dot{u}_i = \sum_j w_{ij}(u_j - u_i) + f_i(t), \qquad M\dot{u} = -L_w u + f(t), \qquad M = \operatorname{diag}(m_i). \tag{9.5}$$

Here w_{ij} has units of capacity per unit time, so that each term is a rate of change of stored amount. With the incidence convention of Section 9.1.4, the oriented flux vector

is CBu and its divergence is $B^TCBu = L_w u$. This is the same balance structure as the finite-volume heat model in Chapter 8.

In the absence of sources, choosing unit capacities gives the equation $\dot{u} = -L_w u$ considered earlier. If capacities are unequal, two vertices with the same incident conductances need not respond equally quickly. In general, we need to assign capacities as part of the model; they do not automatically coincide with vertex degrees. External losses can be included as $-k_i(u_i - u_i^{\mathrm{ext}})$ with $k_i \geq 0$; these exchanges with the environment destroy conservation within the network. In the following, we first consider a closed network with $f = 0$ and no losses.

9.3.2. Conservation, positivity, and equilibration

The first property we can derive is conservation of the total stored amount. Indeed, since $\mathbf{1}^T L_w = 0$, we obtain

$$\frac{d}{dt}(\mathbf{1}^T M u) = 0.$$

At a vertex attaining the current maximum, every difference $u_j - u_i$ is nonpositive; at a minimum the signs are reversed. Consequently the maximum cannot increase and the minimum cannot decrease. In particular, nonnegative initial data remain nonnegative. We can also express this property in matrix form. The matrix $-M^{-1}L_w$ has nonnegative off-diagonal entries and zero row sums, so $e^{-tM^{-1}L_w}$ is a nonnegative row-stochastic matrix for $t \geq 0$. Each value at time t is therefore a convex combination of initial values.

For a connected graph, we define the conserved capacity-weighted mean as

$$\bar{u}_M = \frac{\mathbf{1}^T M u(0)}{\mathbf{1}^T M \mathbf{1}}.$$

The disagreement energy satisfies

$$\frac{d}{dt}\frac{1}{2}\|u - \bar{u}_M \mathbf{1}\|_M^2 = -u^T L_w u = -\sum_{\{i,j\}\in E} w_{ij}(u_i - u_j)^2 \leq 0, \qquad \|x\|_M^2 = x^T M x. \quad (9.6)$$

Thus diffusion conserves the total amount while dissipating differences. Every equilibrium is constant on each connected component, by the Laplacian nullspace characterization. On a connected graph the only possible equilibrium with the given conserved amount is $\bar{u}_M \mathbf{1}$; the spectral argument below proves convergence to it. On a disconnected graph each component preserves its own amount and converges to its own capacity-weighted mean.

Example 9.3.2.1. Let us consider two vertices connected by conductance w. Their values satisfy

$$\frac{d}{dt}(u_1 - u_2) = -w\left(\frac{1}{m_1} + \frac{1}{m_2}\right)(u_1 - u_2).$$

Their difference decays exponentially, while $m_1 u_1 + m_2 u_2$ stays fixed. The limiting value is therefore a weighted mean, which in general differs from the arithmetic mean.

9.3.3. Spectral modes and the time scale of diffusion

To study convergence to equilibrium, we can proceed as in the case of unit capacities. The matrix $M^{-1}L_w$ is not necessarily symmetric. However, it is similar to the symmetric positive semidefinite matrix

$$H = M^{-1/2} L_w M^{-1/2}.$$

Equivalently, we can solve the generalized eigenvalue problem

$$L_w v_r = \nu_r M v_r, \qquad v_r^T M v_s = \delta_{rs}, \qquad 0 = \nu_1 < \nu_2 \le \cdots \le \nu_n$$

for a connected graph with at least two vertices. Expanding the initial condition in this basis gives

$$u(t) = \bar{u}_M \mathbf{1} + \sum_{r=2}^{n} e^{-\nu_r t} v_r v_r^T M u(0).$$

In particular,

$$\|u(t) - \bar{u}_M \mathbf{1}\|_M \le e^{-\nu_2 t} \|u(0) - \bar{u}_M \mathbf{1}\|_M.$$

The time $1/\nu_2$ controls the slowest relaxation, whereas $1/\nu_n$ is the fastest modal time scale. For $M = I$ these are exactly the Laplacian modes described in Section 9.1.4. Weak links between substantial regions can create small ν_2 and a long period of near-equilibrium within regions before global mixing. If ν_n/ν_2 is large, an explicit method needs to resolve the fast modes while following the slow evolution. This is the stiffness phenomenon already encountered in the study of differential equations.

We can also relate this evolution to the random walks considered earlier. For unit capacities and symmetric weights, e^{-tL_w} is also doubly stochastic. Thus the same equation can evolve a column probability vector: from vertex i a continuous-time walker jumps to j at rate w_{ij}, with total departure rate s_i. Its stationary distribution on a connected graph is uniform. Alternatively, we can impose total departure rate one at every vertex: with $P = D_w^{-1}W$ and $D_w = \operatorname{diag}(s_i)$, the generator acting on vertex values is $P - I = -D_w^{-1}L_w$. The corresponding column probabilities satisfy

$$\dot{p} = (P^T - I)p,$$

and their stationary distribution is proportional to s_i, as in the PageRank discussion without teleportation. The vertex-value equation $D_w \dot{u} = -L_w u$ and this probability equation are related by $p = D_w u$, but their conserved sums and interpretations differ. These formulas assume no isolated vertices. Continuous-time diffusion converges even on bipartite graphs: exponential decay removes the alternation of a discrete-time walk that must jump at every step.

This second choice leads to a normalized version of the Laplacian.

Definition 9.3.3.1. The matrix $L_{\mathrm{rw}} = D_w^{-1}L_w = I - P$ is called the *random-walk Laplacian*. Its symmetric counterpart is

$$L_{\mathrm{sym}} = D_w^{-1/2} L_w D_w^{-1/2} = I - D_w^{-1/2} W D_w^{-1/2}.$$

For positive weighted degrees s_i, these matrices are similar: $L_{\mathrm{sym}} = D_w^{1/2} L_{\mathrm{rw}} D_w^{-1/2}$. Their eigenvalues therefore agree and can be computed through a symmetric problem. The zero eigenvector of L_{rw} is $\mathbf{1}$, whereas that of L_{sym} is $D_w^{1/2}\mathbf{1}$ on a connected graph. Choosing $M = D_w$ in the generalized diffusion problem recovers precisely these normalized modes. With this choice, the weights in the conservation law are the weighted degrees s_i. In a physical model, instead, we would assign the capacity matrix using the information provided by the application. Isolated vertices must be treated separately because D_w^{-1} is then undefined.

Example 9.3.3.2. Let us return to G_ε and take unit capacities. The cut between U and Z has weight ε, so (9.3) gives

$$\nu_2 = \lambda_2 \le \frac{(p+q)\varepsilon}{pq}.$$

Thus the slowest relaxation time is at least $pq/((p+q)\varepsilon)$ and becomes arbitrarily large as the joining conductance decreases. Increasing that conductance cannot decrease λ_2, by its Rayleigh-quotient characterization. The unit-cost shortest-path scores, however, remain unchanged. Importance for routes and speed of equilibration answer different questions on exactly the same graph.

9.3.4. Time discretization and sparse computation

The graph already provides a spatial discretization. We can therefore apply the time integration methods of Chapter 2 to (9.5). Let $u^k \approx u(t_k)$, where $t_k = k\Delta t$, and let us first consider $f = 0$. Applying explicit Euler yields

$$u^{k+1} = (I - \Delta t M^{-1} L_w) u^k.$$

In exact arithmetic, this iteration preserves $\mathbf{1}^T M u^k$. Its modal amplification factors are $1 - \Delta t \nu_r$, so stability in the M-norm requires $\Delta t \le 2/\nu_n$. Strict inequality is needed to damp every nonconstant mode; equality can leave the fastest mode oscillating. If we also want to preserve nonnegativity and the maximum principle, we need the additional condition

$$\Delta t \le \min_{i : s_i > 0} \frac{m_i}{s_i}. \tag{9.7}$$

Indeed, each update is then a convex combination of the old vertex value and its neighbours' values. Note that this condition also implies stability, whereas stability alone does not guarantee positivity. An explicit step costs $O(n+m)$ using the sparse Laplacian.

If we use implicit Euler, we obtain the linear system

$$(M + \Delta t L_w) u^{k+1} = M u^k. \tag{9.8}$$

The coefficient matrix is symmetric positive definite, since $x^T(M + \Delta t L_w)x > 0$ for $x \neq 0$. Every modal factor is $(1 + \Delta t \nu_r)^{-1} \in (0, 1]$, so that the method is stable for every positive step and damps all nonconstant modes. It preserves the total stored

amount and the maximum principle for every step size. To verify the latter property, we consider a maximizing vertex of u^{k+1}. The corresponding equation implies $u_i^{k+1} \le u_i^k \le \max_j u_j^k$; the minimum follows similarly. These statements concern the homogeneous closed system. We should keep in mind that stability does not imply accuracy: a large step may still give a poor approximation of the transient. Both Euler methods have first-order global accuracy.

We may also consider the second-order Crank–Nicolson method, which gives

$$(M + \tfrac{1}{2}\Delta t L_w)u^{k+1} = (M - \tfrac{1}{2}\Delta t L_w)u^k.$$

It conserves the same total and is unconditionally stable, but its modal factor $(1 - \Delta t\nu_r/2)/(1 + \Delta t\nu_r/2)$ approaches -1 for fast modes at large steps. Spurious oscillations and loss of positivity can therefore persist despite stability. For this reason, implicit Euler is often a convenient first choice when positivity and strong damping are important.

For constant coefficients and a fixed step, we only need to assemble L_w once. A sparse factorization of $M + \Delta t L_w$ can then be reused at every implicit Euler step. For larger networks, we can use a preconditioned iterative solver, such as conjugate gradients. Its tolerance needs to be small enough that linear solve errors do not dominate the time discretization error or accumulate as a drift in the total amount. We do not need to form a matrix inverse. For a homogeneous problem requiring only a few output times, a Krylov approximation of the action $e^{-tM^{-1}L_w}u(0)$ is an alternative to stepping through every intermediate time; the full dense exponential is unnecessary.

With sources, implicit Euler has right-hand side $Mu^k + \Delta t f(t_{k+1})$. Its total-amount balance becomes

$$\mathbf{1}^T M(u^{k+1} - u^k) = \Delta t\, \mathbf{1}^T f(t_{k+1}).$$

A time-independent source on a closed connected graph admits a steady state only if $\mathbf{1}^T f = 0$; then $L_w u = f$ determines it up to a constant, fixed by the total amount. Nonzero net input instead changes the mean. Prescribing boundary values or environmental losses gives a different balance and can produce a unique steady state with nonzero net input.

We can now make explicit the connection with electrical flows. Let us set $f = b = e_s - e_t$ and use the same conductances in the electrical and diffusion models. A stationary solution of $M\dot{u} = -L_c u + b$ satisfies $L_c u = b$, exactly the potential equation used for current-flow centrality. The source and sink balance, so the total stored amount remains fixed. On a connected graph, if u_* is the solution with that total amount, then $v = u - u_*$ satisfies $M\dot{v} = -L_c v$ with zero capacity-weighted mean; the modal decay above proves $u(t) \to u_*$. Capacities affect relaxation but not the equilibrium edge currents CBu_*. Adding a constant to the potential leaves these currents unchanged. Electrical flow is therefore the steady response to a balanced, continuously imposed source and sink, whereas unforced diffusion relaxes to a constant state.

9.3.5. A numerical experiment and validation checks

On a connected graph, assign conductances and capacities, choose a vertex i_0, and set $u_{i_0}(0) = 1/m_{i_0}$ and all other values to zero. This places one unit of stored amount

at i_0. Plot $u_i(t)$ at several times, using a fixed colour scale, together with the total amount and the energy in (9.6). The expected limiting value is $1/\sum_i m_i$. Use G_ε as a reproducible first example: take $p = 6$, $q = 3$, and compare $\varepsilon = 1$ with $\varepsilon = 0.1$, keeping capacities fixed and placing the initial amount in the smaller cluster. Then repeat on a road graph with explicitly assigned conductances and capacities. Relate the observed mixing times to the computed ν_2 rather than interpreting a high-centrality vertex as sufficient evidence of rapid diffusion.

First verify an implementation against the explicit two-vertex solution. Then check total-amount conservation, nonnegativity, decay of disagreement, and convergence to the correct componentwise mean. Repeat with half the time step: at a fixed final time in the asymptotic error regime, Euler errors should decrease by approximately a factor of two and Crank–Nicolson errors by four. Use a small-graph spectral solution or an accurate exponential action as a reference. Test positivity separately from norm stability, and monitor the residual of each implicit solve. These checks validate the chosen diffusion model and its discretization; they do not establish that diffusion is an accurate model of observed traffic.

9.4. Resilience and network interventions

9.4.1. Performance before and after a closure

Centrality and diffusion describe structure and dynamics on a specified network. To assess an intervention, we need to choose a performance measure and compare its values before and after deletion of a vertex or edge. Route-based measures quantify accessibility losses; changes in the slowest diffusion rate can instead quantify losses in mixing performance. We first consider routes, keeping their costs fixed during the comparison. Let $\mathcal{P}$ be a set of distinct-endpoint origin–destination pairs that are reachable in the original graph. For deletion of edge e, let $\mathcal{P}_e$ contain those pairs still reachable in $G - e$. We can measure the effect of the deletion through the two quantities

$$f_e = 1 - \frac{|\mathcal{P}_e|}{|\mathcal{P}|}, \qquad \Delta_e = \frac{1}{|\mathcal{P}_e|} \sum_{(s,t)\in\mathcal{P}_e} \frac{d_{G-e}(s,t) - d_G(s,t)}{d_G(s,t)}.$$

If no pair remains reachable, the conditional delay Δ_e is undefined; the disconnection fraction remains meaningful. Even with a fixed original sample, conditional means for different closures can involve different surviving pairs. An alternative with a fixed denominator is efficiency,

$$E(G) = \frac{1}{n(n-1)} \sum_{s\neq t} \frac{1}{d_G(s,t)}, \qquad 1/\infty = 0.$$

Deleting an edge cannot improve this efficiency when the remaining costs are held fixed. A congestion model in which costs change with traffic is a different problem.